
175 PTAS
84
mi computer
CURSO PRACTICO DEL ORDENADOR PERSONAL,
EL MICRO Y EL MINIORDENADOR
Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen VII-Fascículo 84

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, 08008 Barcelona
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-067-2 (tomo 7)
84-85822-82-X (obra completa)
Depósito Legal: B. 52-84

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 218508
Impreso en España-Printed in Spain-Agosto 1985

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 19 425 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 429 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S.A. (Paseo de Gracia, 88, 5.°, 08008 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.



Finalizamos nuestra serie sobre educación con una reflexión acerca de cómo la informática puede revolucionar los métodos educativos

El futuro de la informática educativa presenta un agudo contraste con su pasado. Un hardware más barato, un software perfeccionado y el desarrollo tecnológico se están combinando para cambiar el rostro de la educación tal como la conocemos hoy. Lejos de ser el pariente pobre del mercado de consumo, la educación se está manifestando rápidamente como una de las mayores áreas de desarrollo de una nueva tecnología. La causa de esta súbita mutación de la imagen de la informática educativa no es difícil de entrever. En los primeros días de la revolución tecnológica, la educación fue claramente discriminada por las empresas de hardware y software en favor del más lucrativo mercado de consumo. Esta situación se está modificando rápidamente, en la medida en que los productores empiezan a comprender la importancia de vender en las escuelas e inculcar una conciencia de las marcas comerciales entre los niños, que con el tiempo serán consumidores por derecho propio.

El efecto inmediato de esta actitud es el abaratamiento y la mayor disponibilidad de hardware. En 1985 Apple ha iniciado una campaña vendiendo el Macintosh en las escuelas con un descuento del 30 %. Atari ha seguido el camino marcado por Apple diseñando su nueva gama de micros de 32 bits ST teniendo en mente el mercado educativo,

Hacia el siglo XXI

Centro de aprendizaje
Los ordenadores pueden revolucionar nuestro concepto de la educación al colocar al niño en el centro del proceso del aprendizaje y darle el control de los diferentes recursos para la enseñanza que la nueva tecnología pondrá a su disposición. La tecnología "vía satélite" y los sistemas de conexión en red permitirán que el estudiante se comunique con bases de datos de conocimientos de diferentes países y culturas de todo el mundo. Las escuelas podrían ceder paso a "centros de supervisión del aprendizaje", donde los niños se reunirían para analizar su trabajo, la mayor parte del cual se podrá llevar a cabo en el hogar. Los sistemas de video interactivo pueden constituir una base para el desarrollo de software sumamente complejo y se pueden utilizar robots para demostraciones físicas de conceptos abstractos. Además, el estudiante podrá acceder a los sistemas locales de ordenadores centrales, utilizando un micro a modo de terminal inteligente, y un sistema de este tipo podría supervisar el intercambio de datos, la comunicación alumno-alumno, y ofrecer un mayor poder de proceso cuando ello fuera necesario

Alan Adler

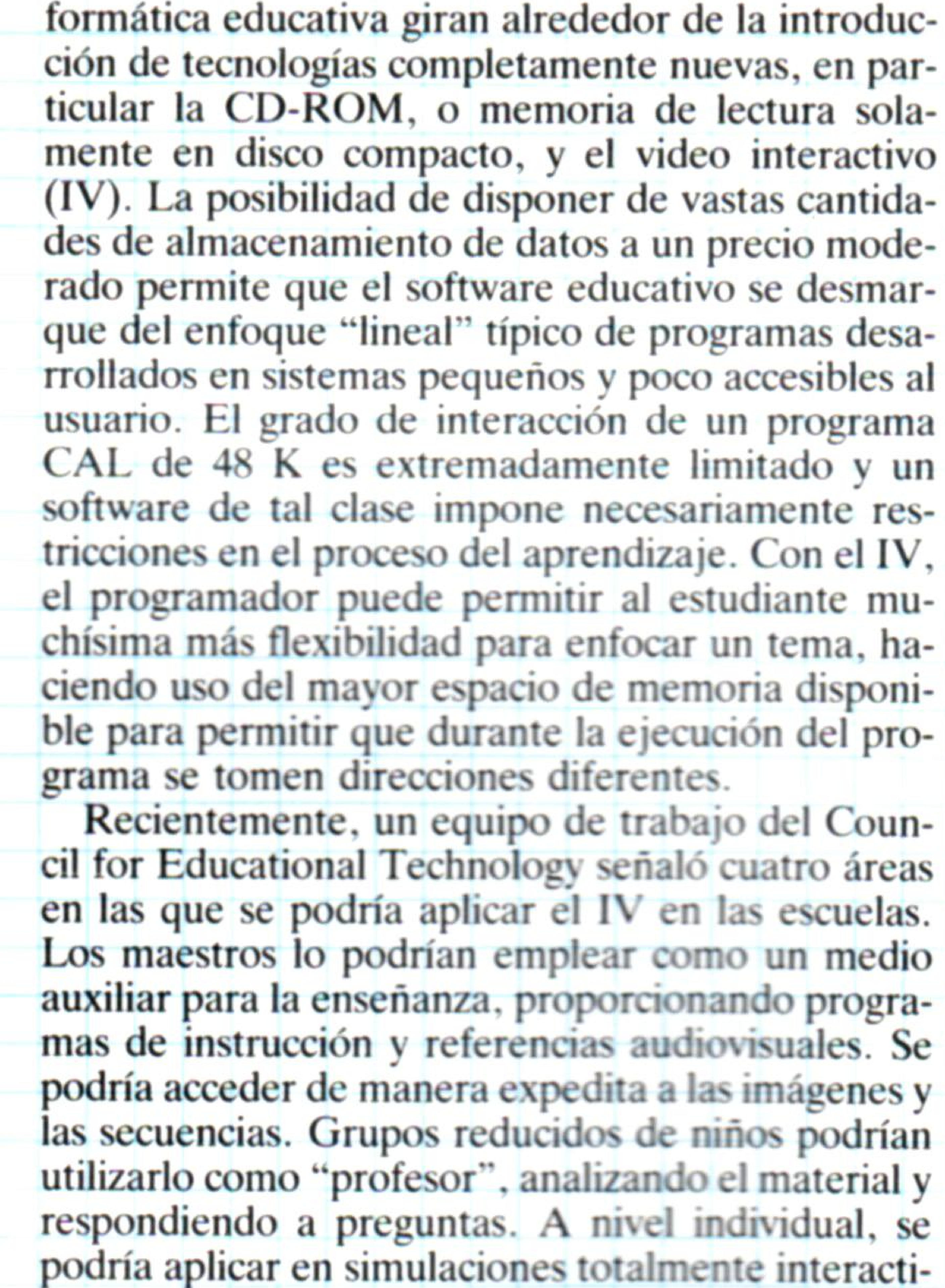
ofreciendo la opción de adquirir una máquina con LOGO en lugar de BASIC, suministrado en ROM.

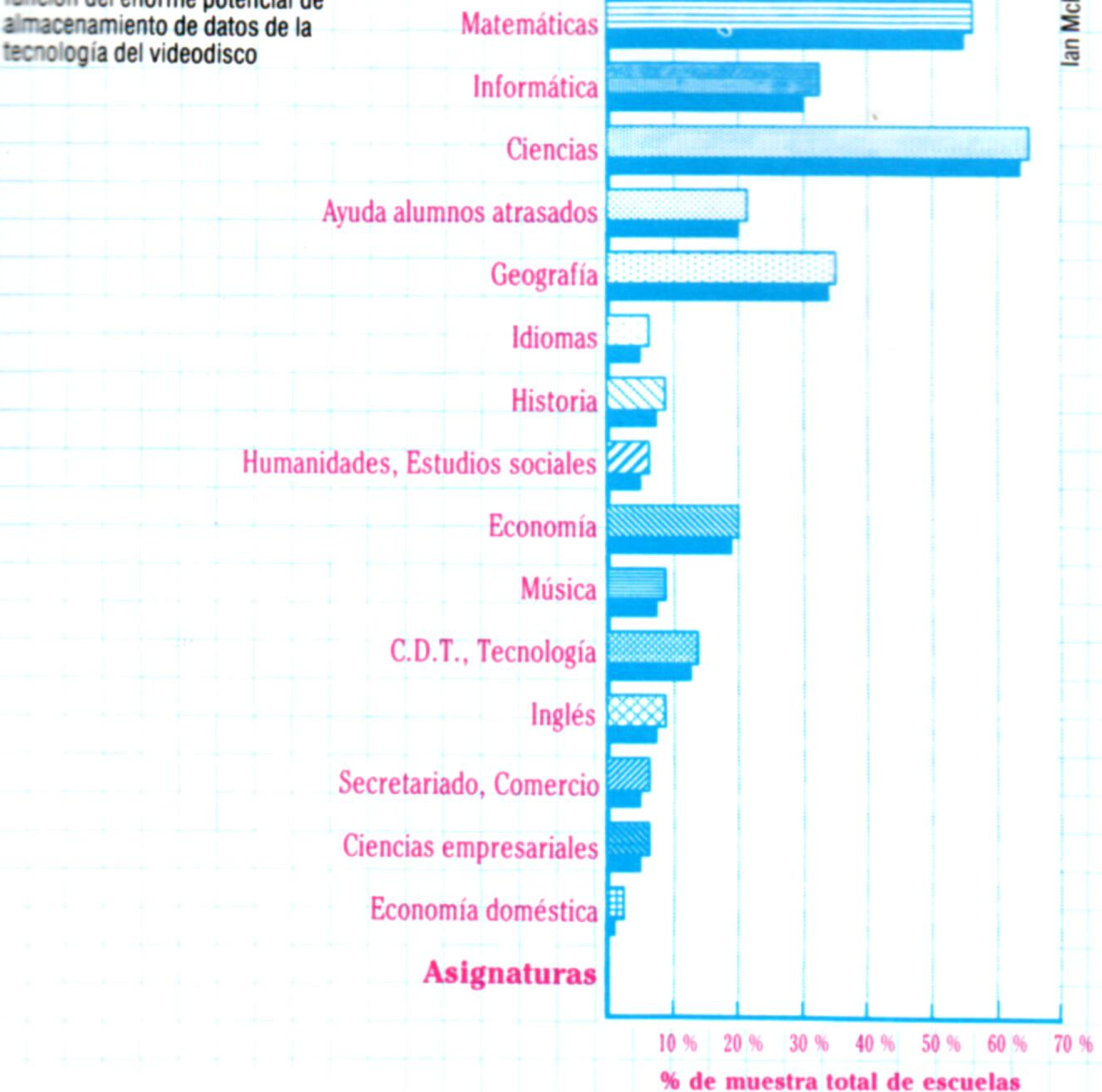
Aparte del sentido comercial de penetrar el mercado educativo, ha surgido, asimismo, una segunda generación de usuarios de ordenadores que demandan usos más serios para sus micros. El desarrollo de software que previamente hubiera estado dirigido de modo exclusivo a las escuelas, de pronto está hallando un mercado más amplio, con lo cual la inversión en software educativo "serio" se convierte en una mejor propuesta comercial. La escasez de hardware y el escaso desarrollo de software ya no constituyen, por lo tanto, un obstáculo grave para la informática educativa; siendo así, ¿qué depara el futuro? El uso de ordenadores en la educación puede ahora avanzar en dos direcciones: la extensión y mejora de las técnicas existentes, y la introducción de métodos totalmente nuevos.

Seymour Papert ya ha señalado algunas ampliaciones que le gustaría introducir en el concepto del LOGO, utilizando el potencial de máquinas como el Macintosh. Papert desea ver escrito en el lenguaje un "microcosmos físico" que permita al niño jugar con las leyes del movimiento de la misma forma con que juega con las leyes de la geometría. Asimismo, prevé un "microcosmos de base de datos" en el que el niño pueda aprender sobre el proceso de la información. Algunas de las ideas de Papert sobre el microcosmos físico ya se han incorporado en el LOGO del Mac. Las simulaciones, los programas CAL y las bases de datos pueden volverse más amables y más potentes en la medida en que programas como el Macintosh Filevision se abran paso en las escuelas.

Sin embargo, las áreas más interesantes de la informática educativa giran alrededor de la introducción de tecnologías completamente nuevas, en particular la CD-ROM, o memoria de lectura solamente en disco compacto, y el video interactivo (IV). La posibilidad de disponer de vastas cantidades de almacenamiento de datos a un precio moderado permite que el software educativo se desmarque del enfoque "lineal" típico de programas desarrollados en sistemas pequeños y poco accesibles al usuario. El grado de interacción de un programa CAL de 48 K es extremadamente limitado y un software de tal clase impone necesariamente restricciones en el proceso del aprendizaje. Con el IV, el programador puede permitir al estudiante muchísima más flexibilidad para enfocar un tema, haciendo uso del mayor espacio de memoria disponible para permitir que durante la ejecución del programa se tomen direcciones diferentes.

Recientemente, un equipo de trabajo del Council for Educational Technology señaló cuatro áreas en las que se podría aplicar el IV en las escuelas. Los maestros lo podrían emplear como un medio auxiliar para la enseñanza, proporcionando programas de instrucción y referencias audiovisuales. Se podría acceder de manera expedita a las imágenes y las secuencias. Grupos reducidos de niños podrían utilizarlo como "profesor", analizando el material y respondiendo a preguntas. A nivel individual, se podría aplicar en simulaciones totalmente interactivas, programas CAL sofisticados y aprendizaje de





Ian McKinnell

La eficacia del video
El uso de micros por departamentos en las escuelas británicas refleja claramente una tendencia muy fuerte hacia las aplicaciones científicas. La introducción de los sistemas de video interactivo y el desarrollo de software IV podrían hacer mucho por reorientar este equilibrio. La enseñanza de historia, por ejemplo, podría experimentar un cambio radical mediante el uso de complejas simulaciones, factibles en función del enorme potencial de almacenamiento de datos de la tecnología del videodisco

Proyecto de envergadura

La enorme capacidad de almacenamiento del videodisco abre posibilidades para aplicaciones completamente nuevas. La BBC, en colaboración con Philips, Thorn-EMI y el Microcomputers in Education Project, está desarrollando a nivel nacional un proyecto para recopilar datos sobre la Gran Bretaña contemporánea. Las escuelas de todo el país reunirán datos relacionados con su entorno inmediato, todos los cuales se compaginarán para formar una "base de datos del pueblo" que contenga detalles sobre el empleo de la tierra, distracciones, servicios, etc. Además, los recursos nacionales de datos (p. ej., el censo y los informes sobre transportes) se combinarán en una inmensa "base de datos nacional", y ambas bases de datos residirán en un único disco láser. El proyecto constituirá un notable documento de la sociedad contemporánea y, gracias a la tecnología del disco láser, contendrá no sólo texto, sino también mapas, diagramas, cuadros e incluso secuencias de imágenes animadas



final abierto. Por último, se podría utilizar para acceder a datos, como una biblioteca de material de referencia audiovisual.

Aunque es necesario un gran trabajo para indexar y acceder a la información y desarrollar software para IV, el sistema ya se está introduciendo en muchas áreas diferentes. La Open University (Universidad Abierta) británica lo ha utilizado en escuelas de verano donde, debido al corto tiempo disponible, los estudiantes de ingeniería tenían dificultades para estudiar el efecto de, por ejemplo, la fatiga en diversos materiales. Se ha escrito un programa IV que emplea fotogramas fijos, animación y secuencias de video. Toma la forma de una batalla en la sala de tribunal entre los fabricantes de ositos de felpa y los proveedores de las arandelas que se utilizan para asegurar los ojos de los ositos. El estudiante debe realizar una serie de experimentos, que conformarán la base de la evidencia utilizada en el tribunal. El programa interactivo se controla mediante un gran programa CAL que "analiza" las posibles causas de defectos de polímero en la substancia que se utiliza para los ojos de los ositos. El estudiante lleva a cabo varios experimentos e interpreta los resultados, que finalmente se presentan en una secuencia dramatizada.

No más escuelas

La transferencia de técnicas de software existentes (simulaciones, CAL, bases de datos, etc.) en sistemas de CD-ROM es apasionante, pero los efectos a largo plazo de la nueva tecnología tienen un alcance mayor. "Yo no creo —afirma Seymour Papert— que de aquí a 20 años existan las escuelas. Es imposible que el concepto de un maestro frente a una clase que toma notas pueda existir en la edad de los ordenadores. Si se repasan una por una todas las cosas que suceden en la escuela, éstas simplemente no serán necesarias, exceptuando aquellas que se dan de forma deficiente, como que los niños hablen entre sí y lleguen a conocerse mutuamente... Existirá algún lugar donde los niños se encuentren y aprendan y se desarrollen. Se llame escuela o no, carece de relevancia. Lo que sí es importante es que la escuela que conocemos en la actualidad ya no existirá."

Sus puntos de vista también los comparte, llevándolos aún más lejos, el profesor Tom Stonier: "El ordenador iniciará el cambio desde la educación basada en la escuela a la educación basada en el hogar. El papel del maestro cambiará, pasando de base de datos andante a consejero de conocimientos."

Las posibles implicaciones de este cambio en la importancia otorgada, de una estructura educativa basada en el sistema a una centrada alrededor del estudiante como individuo, son enormes y no están limitadas de ninguna manera al proceso de aprendizaje. El profesor Stonier es coautor de un libro titulado *Children, computers and communications* (Niños, ordenadores y comunicaciones), en el que predice la educación a través de un sistema de "abuela electrónica".

De acuerdo a Stonier, a medida que la educación pase de la escuela al hogar y las mejoras en el campo de la salud y la medicina incrementen el promedio de vida del hombre, los ciudadanos adultos se irán comprometiendo más en la educación, como en realidad lo hacen ya en muchas de las sociedades llamadas "primitivas". Stonier piensa que los ancianos, uno de nuestros sectores sociales más postergados, están idealmente cualificados para asumir la responsabilidad de una sección del sistema educativo. En Estados Unidos existe ya un ejemplo de cómo podría desarrollarse el sistema. En Portage (Wisconsin) un maestro visita a las familias durante medio día una vez por semana. En esta visita, el maestro evalúa los progresos hechos por el niño durante la semana anterior y decide, en colaboración con los padres, en qué dirección se encaminará el trabajo de la siguiente semana.

Aunque el sistema de Portage se centra en la educación de niños incapacitados, existen todas las posibilidades de que se amplíe a otros estudiantes. En Gran Bretaña es perfectamente legal educar a los hijos en casa si se puede convencer a la autoridad local en materia de educación de que uno es capaz de hacerlo. El número de familias que están optando por salir del sistema es cada vez mayor, y en la actualidad son más de 2 000 las que han "desescolarizado" a sus hijos. El acceso a inmensas bases de datos públicas, los potentes micros con grandes bases de datos de CD-ROM y el software interactivo les ofrecerán a los padres una eficacísima herramienta para educar a sus hijos en el propio hogar.

A medida que aumentan la complejidad y el alcance del software educativo, el papel del maestro se vuelve cada vez menos fácil de definir. Stonier predice que los estudiantes llegarán a estar mejor informados que sus propios maestros. En la medida en que los maestros cumplan con la función de "base de datos caminante", el conocimiento que adquiera un niño se verá severamente limitado por la comprensión del maestro. Todos los profesores de informática, e incluso los de la mayoría de otras asignaturas, pueden señalar un niño que los ha superado. Stonier piensa que este patrón se repetirá a través de todo el plan de estudios a medida que se vayan desarrollando bases de datos y software educativo.





BBC Open University Production Centre

Testigo ocular
Las secuencias cinematográficas y la información visual de alta resolución hacen necesario procesar y almacenar inmensas cantidades de datos; por supuesto, ello es imposible en los sistemas controlados por microordenadores, pero actualmente la tecnología del video interactivo (IV) se está imponiendo con rapidez y a un precio asequible. Estos fotogramas pertenecen a *The teddy bear disc* (El disco del osito de felpa) y muestran un sistema de videodisco controlado por ordenador en plena acción, en este caso un programa sobre materiales de ingeniería para alumnos universitarios. Éste se desarrolla en un tribunal industrial y la tecnología IV permite que el estudiante asuma el papel de "testigo pericial", llevando a cabo experimentos y utilizando el sistema para construir un caso legal

La revolución del CD-ROM

Para almacenar los datos de forma fiable, gran parte del espacio de los CD-ROM debe dedicarse a rutinas de verificación de errores e información de formateo. No obstante, un disco CD-ROM ofrece al usuario 552 megabytes de almacenamiento de datos, lo que permite almacenar en un solo disco millares de pantallas de información. Además, a diferencia de los reproductores de discos CD de audio, que necesitan sofisticados convertidores D/A, los reproductores de CD-ROM sólo han de tratar con datos digitales, y esto simplifica el sistema de circuitos del reproductor y reduce el costo unitario. Teniendo en cuenta que los reproductores, en los otros aspectos, son casi idénticos a las unidades de audio, es probable que los precios permanezcan bajos debido a la demanda del consumidor y las técnicas de fabricación estándar. Los discos de CD-ROM se venderán a precios similares a los de los discos compactos de audio si las ventas son aceptables. Tras su introducción se producirá un auténtico cambio en la comercialización de software

Variables valiosas

Analizamos el uso de funciones y ofrecemos varios ejemplos que ilustran la solidez estructural del PASCAL

Así como los procedimientos se pueden definir y llamar por su nombre, también se puede llamar a las funciones, pero sólo formando parte de expresiones, no como sentencias. Mientras que las llamadas a procedimientos invocan la ejecución de subprogramas para llevar a cabo algún proceso, las llamadas a funciones *calculan* un valor. El resultado devuelto puede ser de cualquier tipo simple, real o escalar, o de tipo *puntero* (que aún no hemos tratado). La única diferencia en el encabezamiento, además de utilizar la palabra reservada FUNCTION, es que se debe especificar el identificador de tipo del resultado devuelto. Supongamos que el PASCAL no dispusiera de la función predefinida impar (*odd*); sería fácil definir una propia:

```
FUNCTION Impar (numero:integer);boolean;
  BEGIN
    Impar:=numero MOD 2>0
  END;{Impar}
```

El valor se devuelve en el identificador de función y, por lo tanto, debe serle asignado a él en el cuerpo de la función, como podemos ver. En este sentido, los nombres de funciones son como variables que no se inicializan nunca, pero sus valores se calculan cada vez que aparecen en una expresión. Sólo pueden aparecer en el lado izquierdo de una asignación, de modo que si fuésemos descuidados y programáramos la asignación como:

```
IF numero MOD 2>0 THEN
  Impar:=true
```

(olvidando la necesaria cláusula ELSE Impar:=false), existe en la construcción un camino que deja el resultado indefinido.

Al igual que con los parámetros de procedimientos que hemos visto hasta ahora, el valor del parámetro actual se pasa al identificador entero local, número, desde el punto de "activación". De modo que:

```
WriteLn (Impar(sqr(N DIV 100)))
```

que siempre imprimiría False, realiza cuatro operaciones:

1. se calcula la expresión N DIV 100
2. este valor entero temporal se pasa a srq como un parámetro de valor actual
3. se devuelve su cuadrado y se le pasa a Impar
4. ahora se calcula el resultado booleano y se pasa al procedimiento WriteLn como otro parámetro de valor.

Si cuando se ejecutara la sentencia de arriba, N tuviera el valor 17, ¿cuál sería después su valor? Ésta puede parecer una pregunta ridícula, por supuesto. No hay ninguna razón posible por la cual el valor de N cambie durante la evaluación de cualquier función que implique a N como un parámetro. Los resultados de todas las funciones deben depender únicamente del "estado del mundo" tal como existe. Es decir, el valor devuelto es una "función" (sic) del (de los) valor(es) de sus "argumentos" o parámetros. El mecanismo del PASCAL para pasar sólo el valor de las variables asegura que, aun cuando los parámetros cambien localmente en virtud de alguna función o algún procedimiento, y debido a que en realidad son copias locales, en el punto de activación los parámetros reales no se verían afectados.

Por estos mismos motivos, aunque el PASCAL no lo prohíbe, los datos no deben ser accedidos globalmente. Todos los enlaces de datos entre llamadas a procedimientos y a funciones se deben controlar mediante listas de parámetros, aunque dichos datos se hallen dentro del ámbito del subprograma. La única excepción a esta regla general es para el acceso a constantes globales. Por su propia naturaleza, éstas no pueden ser dañadas en PASCAL, porque es imposible alterar sus valores. Sin embargo, si se pasara como parámetro un valor de constante, se convertiría en una variable local y, por consiguiente, dejaría de ser inexpugnable.

```
FUNCTION Minuscula (caracter:char)               :char;
  {devuelve la minuscula de cualquier argumento
  mayuscula}
  CONST
    offset=32;{ASCII ord ('a')–ord('A')}
  BEGIN
  IF caracter IN['A'..'Z']
    THEN
      Minuscula:=chr(ord(caracter)+offset)
    ELSE
      Minuscula:=caracter
  END;{Minuscula}
```

Cuando definimos procedimientos, en algunas ocasiones es esencial que *alteren* los valores de sus propios parámetros; de lo contrario, no llevarían a cabo la tarea para que fueron diseñados. Un ejemplo obvio es el propio procedimiento read del PASCAL. No sería muy útil que read (N) diera meramente un valor a un entero que fuera local de read y que no se modificara el valor de N. En tales casos necesitamos pasar la dirección de un parámetro variable (en vez de su valor) con el fin de que el procedimiento se pueda referir directamente a él en lugar de a su copia local. Este mecanismo se denomina *paso por dirección* o *por referencia* y normalmente sólo se debe utilizar con procedimientos y no con funciones.

El paso de parámetros variables se consigue simplemente incluyendo la palabra reservada VAR antes del o de los elementos de la lista de parámetros de procedimiento. Se puede considerar que la sintaxis de una lista de parámetros es idéntica a la declaración VAR de un bloque; pero en vez de que VAR aparezca una sola vez como delimitador, sólo aparece antes de un elemento que necesita ser VARiado por el procedimiento. Por ejemplo:

```
PROCEDURE Proceso (VAR contador : ListaContador);
```



Desarrollo de algunas aplicaciones

Para ilustrar esta potentísima técnica, vamos a desarrollar un programa que lea algunos nombres desde el teclado y, relacionada con cada nombre, una cantidad de dinero que nos debe cada una de esas personas. Por razones de simplicidad, utilizaremos sólo una serie por nombre y expresaremos las deudas como cantidades enteras en pesetas. Una vez probado el programa, podríamos añadir más detalles, como direcciones, números de teléfono, etc. Podríamos imprimir la lista de nombres tanto por orden alfabético ascendente como descendente, o quizá, por el orden de las cantidades adeudadas. La estructura natural de los datos será una lista de registros que contengan campos del tipo apropiado cuya adición resulte fácil posteriormente.

Podemos manipular cada registro como un dato individual y utilizar cualquiera de los campos como "clave" para clasificación. Antes que nada, diseñemos la representación completa de los datos:

```
CONST
   LongSerie    = 20;
   LongLista    = 50;
TYPE
   Cardinal     = 0..MaxInt;
   TamañoSerie = 1..LongSerie;
   serie        = PACKED ARRAY [TamañoSerie]
                                        OF char;
   dato              = RECORD
  {otros campos a        nombre :serie;
   conveniencia}          deuda :Cardinal;
                     END;{dato}
   limites         =1..LongLista;
   ListaRegistro =ARRAY [limites] OF dato;
```

Aquí se permiten cincuenta nombres y se da por sentado (de momento) que ningún nombre excederá de 20 caracteres, pero esto lo podemos modificar fácilmente.

Hemos expresado todos los subrangos y estructuras en una parte de definición de TYPE por varios motivos:

1. seguridad: un índice de tipo limites no puede exceder nunca los límites definidos de la matriz, p. ej.
2. localización de errores: en caso de que durante la comprobación se produjera un "error de rango", el mensaje de error ayudaría a localizar el problema.
3. conveniencia y eficacia: las variables locales pueden tener tipos ya existentes, evitando la duplicación y ahorrando memoria.
4. necesidad: el PASCAL insiste en que los "denotadores" de tipo de parámetros deben ser identificadores, no "literales" (0..255, p. ej.).

A continuación, hemos de pensar en unas pocas variables esenciales y en un algoritmo, todo lo cual lo analizaremos en el siguiente capítulo de esta serie. Mientras tanto, pruebe de desarrollar su propio "algoritmo informal" para el programa y confeccione una lista de todos los procedimientos que crea que puedan ser necesarios.

Ventaja positiva

El programa *LectorFichas* manipula números positivos de una forma totalmente segura al leer los datos como caracteres. La conversión al valor numérico equivalente se efectúa restándole el valor original del carácter del dígito al del dígito "0". Esto funciona para todos los juegos de caracteres, porque los caracteres "0".."9" se definen como contiguos. Observe que el programa posee sólo dos variables globales (en el "nivel 0"). Los datos para procesar los números y la función Valor que lleva a cabo la asociación, los utiliza sólo LeerFicha y, por consiguiente, todo ello está declarado y definido dentro de este procedimiento. Tal como está ahora el programa, sólo tiene un problema: si se entrara un valor mayor que la cardinalidad de los enteros del compilador (MaxInt), la sentencia de asignación del bucle WHILE quedaría colgada. ¿Qué le parece la idea de escribir otro procedimiento LeerInt para hacer frente a números negativos?

```
PROGRAM          LectorFichas     (input, output);

TYPE
          cardinal              = 0..MaxInt;

VAR       PosNUm    : cardinal;
          okay      : boolean;

PROCEDURE     LeerFicha     ( VAR  N       : cardinal;
                              VAR  OK      : boolean);
  CONST
          espacio      = ' ';
  TYPE
          simple       =0.9;
  VAR
          simbolo    :char;
          digitos    :SET OF char;
             {2222222222222222222222}
                   {nivel 2}
  FUNCTION  Valor ( digito   : char )      :simple;
          {devuelve el valor numerico de
          un digito en cualquier juego de caracteres}
    BEGIN
    Valor := ord (digito) - ord ('0')

    END;  {Valor}
          {2222222222222222222222}
                          {vuelta a nivel 1}
  BEGIN {LeerFicha}

    REPEAT
      read (simbolo)

    UNTIL simbolo>espacio;

    digitos :=['0'..'9'];

    OK := simbolo IN digitos;

    IF OK THEN
      BEGIN
        N := 0;

        WHILE simbolo IN digitos DO
          BEGIN {calcular en base 10}
            N := 10*N+Valor (simbolo);
            read (simbolo)
          END  {se suprime cualquier car. delimitador}
      END      {o 'EoLn' despues del numero}

  END;      {LeerFicha}
            {1111111111111111111111111111}
BEGIN       {LectorFichas - Nivel 0 Programa Principal}

  page (output);
  WriteLn;
  WriteLn ('Este programa lee enteros positivos');
  WriteLn ('(en la escala 0..',MaxInt:1,')');
  WriteLn ('La terminación del programa se');
  WriteLn ('controla mediante deteccion de error.');
  WriteLn;
  write ('Entre un numero:');
  LeerFicha (PosNum, okay);

  WHILE okay DO
    BEGIN
      WriteLn;
      write ('El numero que entro era:');
      WriteLn (PosNum:1);
      write ('Numero?');
      LeerFicha (PosNum, okay)
    END;

  WriteLn ('—— DETECTADO ERROR ——':40)

END.
```

Entrada doble

Presentamos distintos enfoques de organización de ficheros, desde la base de datos "plana" directa hasta la compleja y poderosa "dBase II"

Todos los registros contenidos en una base de datos "plana" suelen ser totalmente autocontenidos; es decir, cualquier registro dado normalmente retiene toda la información requerida, sin necesidad de referirse a otro registro. Los archivos de fichas tradicionales son planos en este sentido. Podemos pensar, por ejemplo, en una base de datos de biblioteca: cada registro se compone simplemente de los campos TITULO, AUTOR, EDITORIAL e ISBN y no se requiere ninguna referencia cruzada.

Algunas veces, un campo contiene una entrada que puede tener un registro propio. En el archivo de los socios de un club, por ejemplo, un campo HIJOS se puede referir a otros registros del mismo archivo; si los hijos de un socio también son socios del club, ellos tendrán sus propios registros. Pero un campo se puede referir a registros que puedan no tener cabida en la estructura general del archivo de la base de datos. Consideremos una base de datos de inventario de componentes, que contiene los campos NUMERO DE COMPONENTE, PRECIO, CANTIDAD EN STOCK y PROVEEDOR. En este caso, es muy probable que el campo PROVEEDOR también se refiera a otros registros que no responden a la estructura definida para la base de datos de COMPONENTES. Para ilustrar esto veamos, en primer lugar, el archivo SOCIOS DEL CLUB:

NOMBRE DEL SOCIO	Susana Gómez
AÑO DE INSCRIPCION	1979
SUSCRIPCION?	Pagada
OCUPACION	Maestra
SALARIO	89000
HIJOS	Ana Gómez, José Gómez

En este ejemplo, el campo HIJOS contiene dos entradas; ambas podrían tener entradas similares en el archivo SOCIOS DEL CLUB, en el caso de que también fueran socios. Comparémoslo con un registro de la base de datos COMPONENTES:

NUMERO DE COMPONENTE	3995
CANTIDAD EN STOCK	86
PRECIO	34.75
DESCRIPCION	tubo de cobre de 2 m
PROVEEDOR	Widgerama Ltd; Dongle Corp de Taiwan

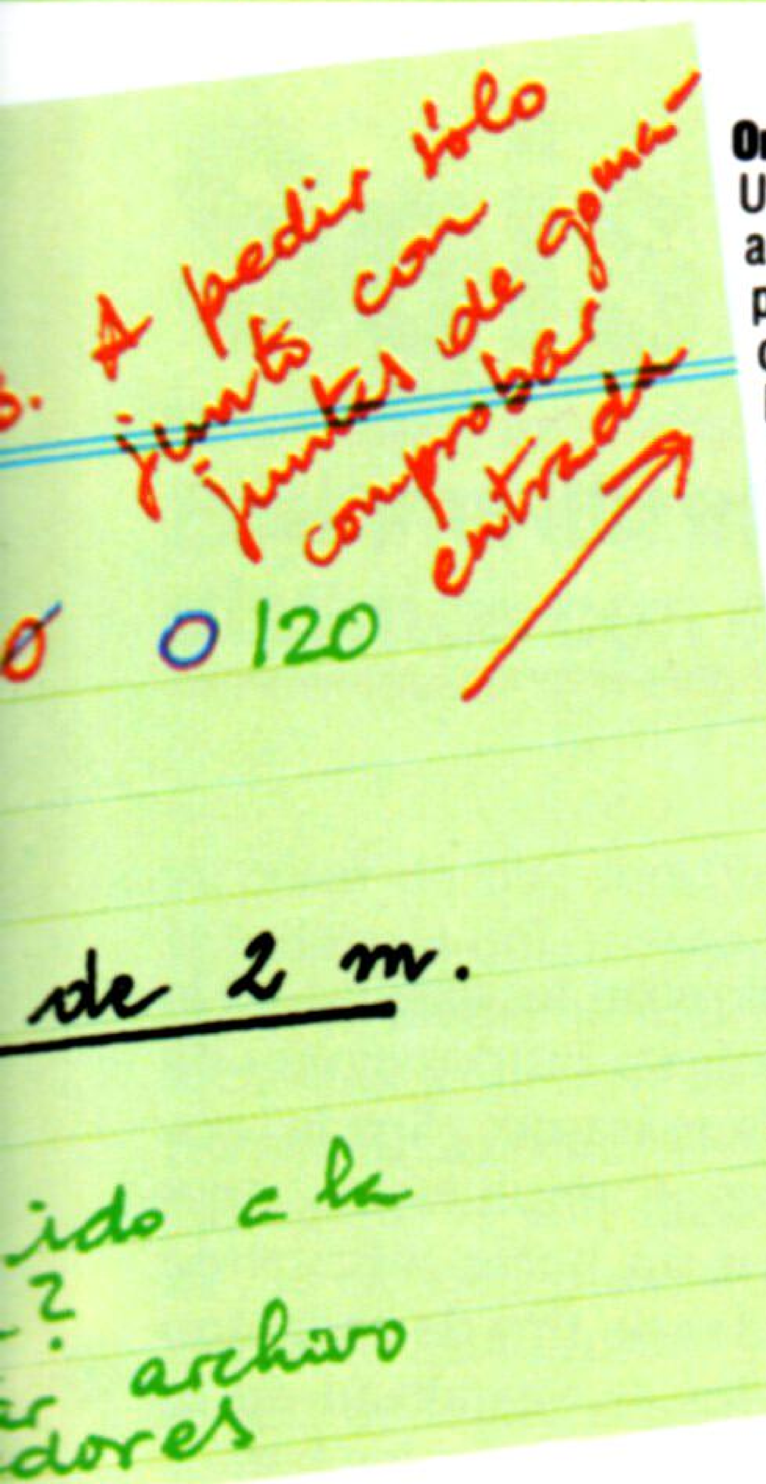
En el campo PROVEEDOR hay dos entradas; ninguna de ellas podría tener registros que se adecuaran a la estructura de la base de datos COMPONENTES. Por consiguiente, un inventario de componentes como éste probablemente tuviera un segundo archivo de base de datos para proveedores, en el cual cada entrada podría ser parecida a la siguiente:

PROVEEDOR	Dongle Corp de Taiwan
DIRECCION	57 Kau Moo Road, Taipei, Taiwan, R.O.C.
TELEFONO	010-886-2-223-4478
SUMINISTRO1	tubo de cobre de 2 m
PRECIO1 ($USA)	34.75
SUMINISTRO2	Pasta para válvulas (lata de ½ litro)
PRECIO2 ($USA)	6.00
SUMINISTRO3	Cera Dazzlebrite (bote pequeño)
PRECIO3 ($USA)	2.30
SUMINISTRO4	Cera Dazzlebrite (bote grande)
PRECIO4 ($USA)	4.00
SUMINISTRO5	—

Es totalmente evidente que la información que necesitamos sobre cada uno de los proveedores es bien diferente, en estructura, de la información que necesitamos sobre los componentes que vende la empresa. La solución consiste en tener dos archivos diferentes: uno para los componentes y otro, separado, para cada uno de los proveedores.

Para implementar con éxito archivos separados pero relacionados entre sí, obviamente es necesario que el DBM sea capaz de manipular más de un archivo a la vez. Los DBM menos sofisticados, como el *Card Box*, del cual ya hemos hablado anteriormente, trabajan sólo con un archivo a la vez, pero los DBM como el *dBase II* pueden usar un archivo como archivo primario (como el de COMPONENTES) y otro archivo (como PROVEEDORES) como archivo secundario. Un buen DBM normalmente permite ejecutar una clasificación en un archivo primario (para extraer registros seleccionados) así como la extracción de registros relevantes (como PROVEEDORES) de archivos relacionados.

El paquete *dBase II* permite utilizar dos archivos relacionados al mismo tiempo, y los asocia median-



Orden de stock
Utilizando su propio sistema de archivo de tarjetas, las entradas pueden ser tan complejas (o confusas) como deseemos. Los DBM deben adoptar un enfoque más estructurado a la entrada de datos, pero los programas sofisticados permiten que el usuario llame a archivos relacionados y, al entrar los datos, se salte ciertos campos en caso de que éstos sean innecesarios en virtud de entradas previas del registro. En el ejemplo, el campo "Enviado nuevo pedido" sólo es relevante si el stock se aproxima a cero o ya lo es. En consecuencia, algunos DBM, como el *Rescue*, permiten que el usuario prepare campos que sean dependientes de campos anteriores o bien que se calculen en función de ellos

te el empleo de campos clave comunes. Dos instrucciones permiten pasar la referencia de uno a otro: SELECT PRIMARY y SELECT SECONDARY (seleccionar primario y seleccionar secundario). Si el archivo primario es COMPONENTES, el secundario, PROVEEDOR, y estuviéramos trabajando en el archivo COMPONENTES, impartiríamos la instrucción USE COMPONENTES. Si después quisiéramos realizar una referencia cruzada al archivo de proveedores, impartiríamos las instrucciones: SELECT SECONDARY y luego, en la siguiente línea, USE PROVEEDOR. Se pueden utilizar todas las instrucciones normales del *dBase II*, que operarán en el archivo que se haya seleccionado más recientemente.

Existen ocasiones en las que el contenido de un campo (como PROVEEDOR en el archivo COMPONENTES mencionado previamente) no requiere las facilidades de registros completamente dependientes (en archivos independientes), de modo que bastarán los campos dependientes dentro del mismo registro. A modo de ejemplo, supongamos que usted lleva una base de datos de la última tecnología en materia de automóviles, almacenando registros de recortes de prensa y referencias a libros que ha visto, relacionados con cosas tan diversas como cuerpos de plástico moldeados a inyección e inyección de combustible controlada electrónicamente, usted podría diseñar un formato como éste:

TEMA: ____________
RESUMEN1: ____________
RESUMEN2: ____________
RESUMEN3: ____________
FUENTE: ____________ ISBN: ____________
TITULO: ____________
FECHA: ____________
N. PAGINA: ____________

Si la fuente de una entrada es una revista, como *Muy Interesante* o *Dirigido por...*, el campo de ISBN será irrelevante, dado que las revistas no poseen ISBN. No obstante, si la fuente de la entrada es un libro llamado *Hacia la comprensión de la informática*, habrá un ISBN. Del mismo modo, si la fuente es un libro o una revista, habrá un número de página. Pero si la fuente fue un programa de televisión, el campo N. PAGINA no será necesario.

Algunos DBM, como el *Rescue*, de Microcomputer Business Systems, permiten que la presencia o la ausencia de cualquier campo en un registro sea calculada o dependa de otros campos previos.

Por consiguiente, si la FUENTE no es un libro, no se solicitará una entrada para el campo ISBN, que no se visualizará (ni en la pantalla ni en las salidas impresas). Si la fuente no es ni un libro ni una revista, el campo N. PAGINA no será necesario y tampoco se visualizará. Un DBM capaz de hacer esto economiza memoria al eliminar los campos innecesarios y ayuda a mejorar la presentación de la información. Será, no obstante, menos versátil que un DBM capaz de relacionar registros de un archivo con registros de otro.

En las bases de datos en las que existe alguna información raíz que permanece constante para todos los registros, y alguna información de ramificación que puede o no ser requerida, tenemos lo que se conoce como *jerarquía de dos niveles*. Como regla general, los DBM de archivos múltiples pueden tratar datos jerárquicos de dos niveles como un conjunto o caso especial de una base de datos de archivos múltiples. Por ejemplo, una referencia FUENTE a un libro podría relacionarse con un archivo LIBROS y una referencia FUENTE a un programa podría relacionarse con un archivo PROGRAMAS.

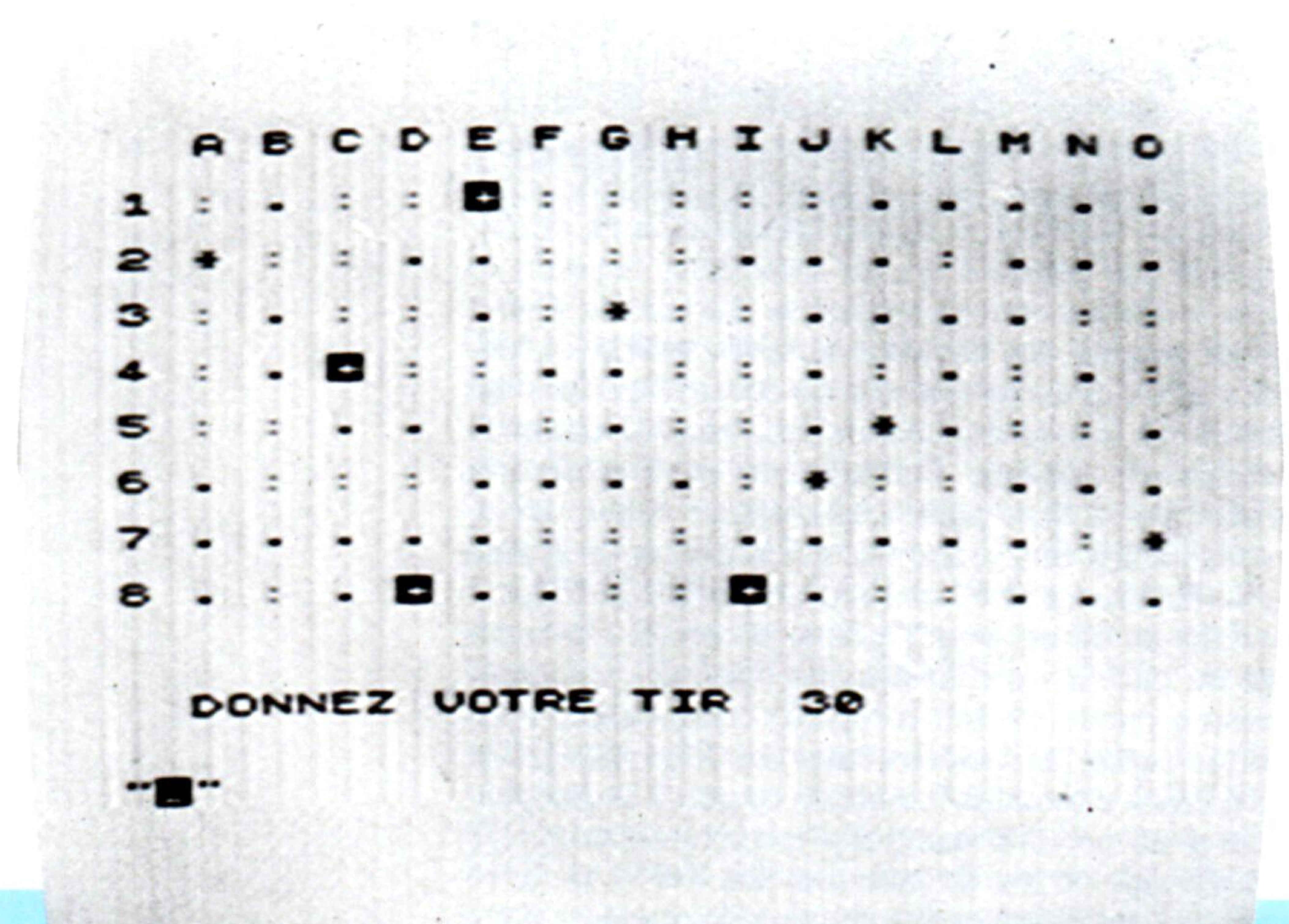
Hay un último punto a tener presente. Un archivo de tarjetas escritas a mano diseñado por usted mismo puede ser tan complejo como desee. Pero habrá serias limitaciones entre los diferentes DBM disponibles comercialmente en cuanto a lo que puedan y no puedan manipular estructuralmente. De modo que antes de adquirir un DBM para su micro, piense qué es exactamente lo que desea que haga y examine con atención las especificaciones para el producto que esté considerando.

Bases de datos al detalle

NOMBRE	FABRICANTE	MAQUINA	FORMATO		OBSERVACIONES
Betabase	Clares	BBC	disco		Long. máx. registro 2048. Campos máx. 200. Tamaño máx. arch. 99 K (40T), 199 K (80T)
Database	Gemini Marketing	Spectrum	cassette		Sistema índice de tarjetas. Regs. definidos por usuario.
DFM Database	Dialog	C64	disco		15 campos/registro. Campo alfanumérico de 36 caracteres. Campo numérico de 9 cifras.
Practifile	Computer Software Associates	C64	disco		3800 reg/archivo. Reacomoda datos para lista de correspondencia. Entrada remota.
MicroPen	AMSoft	Amstrad	disco		Contiene procesador de textos y hoja electrónica incorporados. Long. máx. registro 1024. Núm. máx. registros por archivo:32750.

Batalla naval

Este juego le presenta batalla a su ZX81. Tal vez le sea útil ampliar la memoria a 16 K



Al iniciarse el programa, aparecen las casillas en la pantalla, y el ordenador le solicita las posiciones de sus navíos. A continuación la máquina elige las posiciones de sus propios barcos. Cumplida esta operación, el programa entra en un bucle solicitando vuelta a vuelta las coordenadas de tiro de cada uno de los adversarios. Sus navíos se visualizan en la pantalla bajo la forma de un asterisco, y los barcos hundidos, bajo la de un asterisco invertido. Los sectores alcanzados están representados por ":", y los aún intactos, por ".". El juego finaliza cuando uno de los adversarios ha perdido todos sus barcos. A medida que el juego avanza, el ordenador trabaja de manera más lenta. Es muy importante seguir correctamente el listado al digitar. Para modificar el número de navíos basta cambiar el número 7 en las líneas 250, 330, 401, 460 y 552. Una partida dura entre 25 y 40 minutos.

```
10 REM BATALLA NAVAL
20 REM SALTAR SUBPROGRAMAS
30 GOTO 100
40 LET X=(((((CODE A$(1))-29)+2
   )-1)*2
50 LET Y=(((((CODE A$(2))-38)+2
   )-1)*2)
60 RETURN
70 LET T=PEEK (PEEK 16398+256*
   PEEK 16399)
80 RETURN
100 REM FIJACION
110 PRINT ".";
120 FOR I=38 TO 52
130 PRINT "  ";CHR$ (I);
140 NEXT I
150 PRINT
160 PRINT
170 FOR I=29 TO 36
180 PRINT CHR$(I);"  ";
185 FOR J=1 TO 15
190 PRINT ",  ";
200 NEXT J
204 PRINT
205 PRINT
210 NEXT I
211 RAND
220 PRINT AT 20,0;"COLOQUE SUS
    BARCOS,"
230 FOR K=1 TO 60
240 NEXT K
250 FOR I=1 TO 7
260 PRINT AT 20,0;"COORDENADAS
    DEL BARCO  ";I
270 INPUT A$
275 IF LEN A$<>2 THEN GOTO 270
280 IF CODE A$(1) <29 OR CODE A$
    (1)>36 OR CODE A$(2) <38 OR CODE
    A$(2) >52 THEN GOTO 270
290 GOSUB 40
300 PRINT AT X,Y;
302 GOSUB 70
304 IF T=CODE "*" THEN GOTO 270
05 PRINT "*"
10 NEXT I
20 PRINT AT 20,0;"ESCOGÍ MIS
   POSICIONES..."
330 DIM S$(7,2)
340 FOR I=1 TO 7
350 LET A$=CHR$ (INT (RND*8)+29
    )+CHR$ (INT (RND*14)+38)
360 GOSUB 40
365 PRINT AT X,Y;
370 GOSUB 70
380 IF T=CODE "*" THEN GOTO 350
390 LET S$(I)=A$
400 NEXT I
401 FOR J=1 TO 7
402 FOR K=1 TO 7
403 IF J=K THEN GOTO 405
404 IF S$(J)=S$(K) THEN GOTO 340
405 NEXT K
406 NEXT J
410 LET MS=7
420 LET YS=7
421 LET N=0
425 LET N=N+1
430 PRINT AT 20,0;"  DE SU TIRO
    ";N;"
440 INPUT A$
445 IF LEN A$<>2 THEN GOTO 440
450 IF CODE A$(1) <29 OR CODE A$
    (1) >36 OR CODE A$(2) <38 OR CODE
    A$(2) >52 THEN GOTO 440
460 FOR I=1 TO 7
470 IF A$=S$(I) THEN GOTO 700
480 NEXT I
490 GOSUB 40
500 PRINT AT X,Y;
510 GOSUB 70
520 IF T<>CODE "." THEN GOTO 440
530 PRINT ":";AT 20,0;A$(1);"  "
    ,A$(2);"        RATE.
535 FOR I=1 TO 50
536 NEXT I
540 PRINT AT 20,0;"YO TIRO    ";
    N;"
550 LET A$=CHR$ (INT (RND*8)+29
    )+CHR$ (INT (RND*14)+38)
552 FOR H=1 TO 7
553 IF A$=S$(H) THEN GOTO 550
554 NEXT H
560 GOSUB 40
570 PRINT AT X,Y;
580 GOSUB 70
590 IF T=CODE ":" OR T=CODE "█"
    THEN GOTO 550
600 IF T=CODE "*" THEN GOTO 650
610 PRINT ":";AT 20,0;A$(1);"  "
    A$(2);"          RATE.
620 FOR I=1 TO 50
630 NEXT I
640 GOTO 425
650 PRINT AT X,Y;"█"
651 PRINT AT 20,0;"HUNDIDO.QUEDAN";
660 LET YS=YS-1
670 PRINT YS
675 IF YS=0 THEN GOTO 740
680 FOR I=1 TO 100
690 NEXT I
695 GOTO 425
700 GOSUB 40
701 PRINT AT X,Y;"█"
705 PRINT AT 20,0;"HUNDIDO.QUEDAN";
710 LET MS=MS-1
720 PRINT MS
725 IF MS=0 THEN GOTO 740
730 GOTO 675
740 IF YS=0 THEN LET W$="HE"
750 IF YS=0 THEN LET L$="HA"
760 IF MS=0 THEN LET W$="HA"
770 IF MS=0 THEN LET L$="HE"
780 FOR I=1 TO 20
800 NEXT I
810 CLS
820 PRINT W$;" GANADO."
830 PRINT
840 PRINT L$;" PERDIDO."
850 STOP
```



Seis robots

Esta vez examinaremos un "kit" que permite construir seis robots distintos con el mismo conjunto de piezas

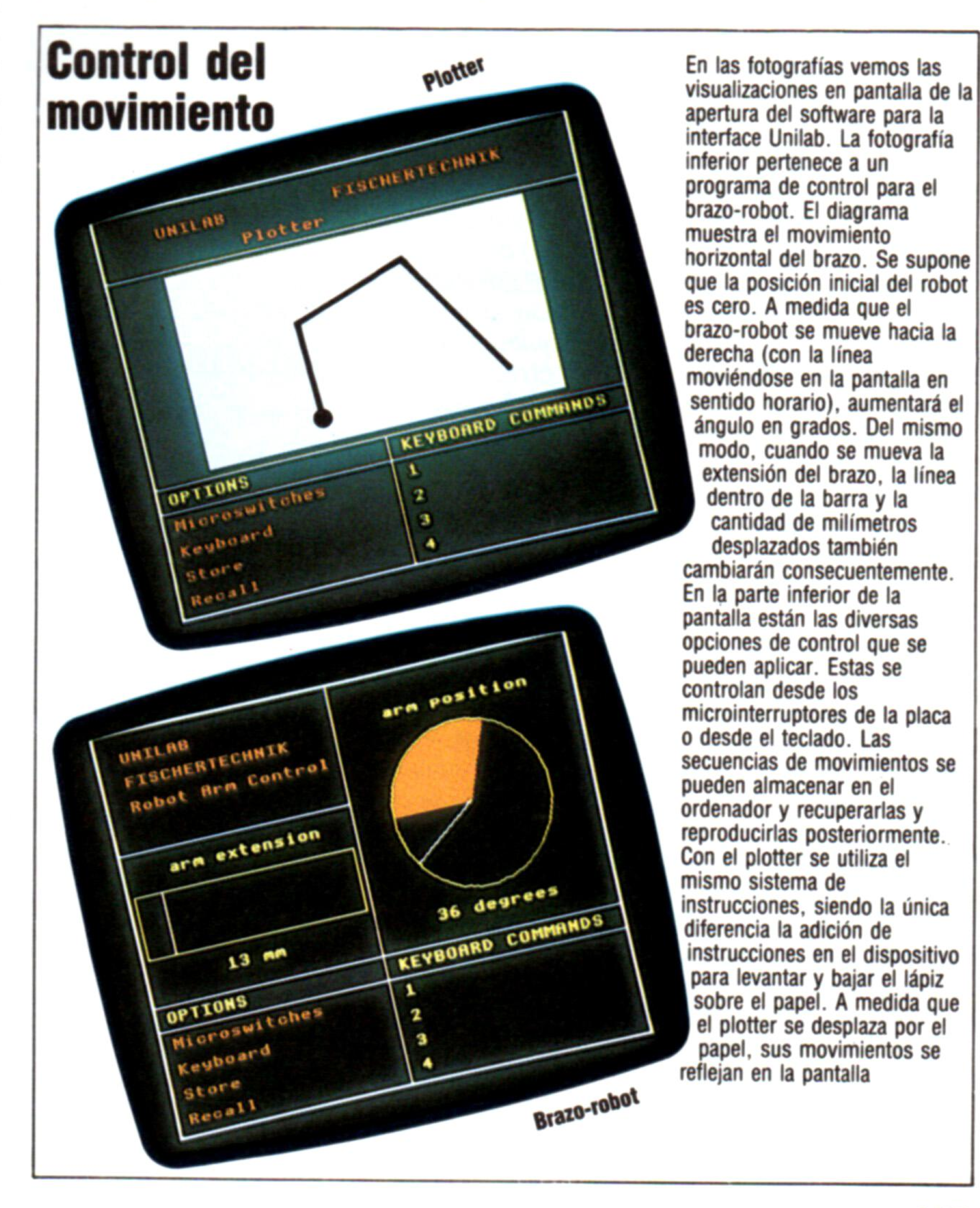
A pesar de que muchas personas relacionadas con la industria del ordenador personal han aventurado durante algún tiempo que la próxima área de gran crecimiento dentro de la industria sería la robótica personal, las predicciones aún no son una realidad. Los motivos son fáciles de detectar. A pesar del hecho de que durante el último año han salido al mercado numerosos "robots personales", han adolecido de algunos problemas graves que han impedido que el público se convenza de que una introducción a la robótica es una actividad interesante y valiosa.

Por otra parte, están los robots de construcción sencilla, como los Movits, que los puede montar en pocas horas cualquier persona que no tenga ningún conocimiento de robótica e incluso ni siquiera de electrónica. En este caso el problema es que estos dispositivos tienen unas aplicaciones muy limitadas y probablemente el usuario pierda enseguida el interés por las máquinas. Por otra parte, existen robots más caros que por lo general exigen un amplio conocimiento de robótica y electrónica para poder manejarlos. Aun así, muchos de estos robots más avanzados poseen sólo una única finalidad. Nuevamente, esto significa que, tras haber explorado todas las rutas, el robot quedará olvidado acumulando polvo en un estante.

Por lo tanto, parece obvio que lo que se requiere es un robot que sea fácil de construir y posea numerosas funciones diferentes que puedan ser exploradas por el usuario. Esto es lo que se pretende con el Fischertechnik Robotics Kit. La idea que ha servido para crear este *kit* es bastante simple. Ya existen desde hace muchos años numerosos juegos para armar electrónicos, dirigidos a los niños, que les permiten realizar circuitos sencillos para construir, por ejemplo, una radio básica. Este mismo *kit* se puede luego desmontar y volver a componer para crear un sencillo dispositivo temporizador electrónico o una alarma antirrobo. Simultáneamente, generaciones de niños han crecido con juegos de construcción Lego que se pueden utilizar una y otra vez para construir una enorme cantidad de modelos diferentes a partir de los mismos bloques plásticos. Al diseñar sus *kits*, Fischertechnik ha adoptado estas dos ideas y las ha combinado con la moderna tecnología del ordenador para desarrollar un producto que bien podría convertirse en el punto de penetración popular para la robótica que tanta gente ha estado prediciendo.

El *kit* Robotics permite que el usuario construya varios diferentes dispositivos robot que se pueden hacer funcionar desde un micro personal. Estos robots van desde un brazo-robot y una máquina que clasifica por orden elementos de longitudes diversas, hasta un plotter y un dispositivo para entrada de gráficos. Una vez que el usuario ha experimentado con un robot determinado, lo puede desarmar y reconstruir para hacer otro. Para poder operar los robots desde un ordenador, se requiere una interface para convertir las señales digitales del ordenador en aquellas que se puedan utilizar para operar los motores eléctricos del *kit*.

Los componentes del juego de construcción constan de varias piezas de plástico y se pueden unir entre sí (al estilo de un juego Lego) para formar diversos modelos. Estas piezas tienen forma de bloques que se pueden unir entre sí para formar las piezas más largas de los brazos, o utilizar para sostener las unidades de control. Otras piezas para construir las unidades mecánicas son ruedas dentadas y tornillos giratorios para hacer girar los dientes. También hay un gran tablero plástico que mide 260×187 mm, que se utiliza ya sea como base para el propio robot o bien, en caso de que se incorpore el plotter o la tablilla para gráficos, como la base para el papel. El sistema de potencia y control eléc-



Control del movimiento

En las fotografías vemos las visualizaciones en pantalla de la apertura del software para la interface Unilab. La fotografía inferior pertenece a un programa de control para el brazo-robot. El diagrama muestra el movimiento horizontal del brazo. Se supone que la posición inicial del robot es cero. A medida que el brazo-robot se mueve hacia la derecha (con la línea moviéndose en la pantalla en sentido horario), aumentará el ángulo en grados. Del mismo modo, cuando se mueva la extensión del brazo, la línea dentro de la barra y la cantidad de milímetros desplazados también cambiarán consecuentemente. En la parte inferior de la pantalla están las diversas opciones de control que se pueden aplicar. Estas se controlan desde los microinterruptores de la placa o desde el teclado. Las secuencias de movimientos se pueden almacenar en el ordenador y recuperarlas y reproducirlas posteriormente. Con el plotter se utiliza el mismo sistema de instrucciones, siendo la única diferencia la adición de instrucciones en el dispositivo para levantar y bajar el lápiz sobre el papel. A medida que el plotter se desplaza por el papel, sus movimientos se reflejan en la pantalla

trico también se debe ensamblar a partir de componentes separados. El juego incluye diversos conectores macho y hembra, ocho dispositivos de conmutación y un par de motores eléctricos. Se proporciona, asimismo, un metro de cable plano de 20 vías y un pequeño trozo extra de cable para realizar "parches".

En general, las piezas son de construcción sólida y tienen aspecto de poder ofrecer muchos años de servicio. La única excepción en este sentido parecen ser los potenciómetros. La placa base de uno de ellos se rompió durante la construcción y, si bien esto no incidió en el funcionamiento del brazo (que vemos en la fotografía) porque la placa base se podía colocar a presión en su sitio, sí plantea una duda sobre su fiabilidad a largo plazo. Otro punto en relación a los potenciómetros es que, a diferencia del resto del cableado, que se puede atornillar en conectores, no hay ningún método para fijarles los cables pelados como no sea envolviéndolos alrededor de los conectores. En este caso es aconsejable que el aficionado que pretenda darle un gran uso al *kit*, suelde las junturas de modo permanente para evitar cortocircuitos que podrían dañar los potenciómetros.

Instalación de los componentes

Construir los robots es bastante sencillo y las piezas en su mayoría son suficientemente grandes como para que las puedan manipular los dedos menos diestros. El principal problema que se plantea en la construcción es el manual, que probablemente sea el punto más débil de todo el sistema. Si bien posee algunas partes estimables (p. ej., las conexiones del circuito son fáciles de seguir), se sustenta en gran medida en fotografías de poca calidad de los modelos parcialmente construidos. Junto con vistas esquemáticas del robot en construcción se muestran fotografías de las piezas que se requieren. El problema es que estas fotografías no ofrecen grandes detalles y carecen de comentarios, por lo cual a menudo los usuarios se encontrarán escudriñando una fotografía para tratar de descubrir dónde se supone que debe ir una pieza determinada. Además, como las fotos muestran al robot en una sola dirección, con frecuencia uno tiene que adivinar dónde está instalada una pieza cuando su posición no aparece a la vista.

No obstante, tal vez lo más grave sea que las fotos son en blanco y negro. Esto significa que cuando finalmente uno llega a la parte final, que es instalar el cableado del circuito en el robot, es casi imposible dilucidar el procedimiento de cableado correcto. Esto es particularmente extraño si se considera que el circuito depende en gran medida de la codificación por colores del cable plano. Los escuetos comentarios que se ofrecen en los diagramas de cableado tampoco son de gran ayuda, diciendo, por ejemplo, que los cables de E3 a E8 pertenecen a una sección determinada.

Dicho esto, hemos de señalar que el robot que presentamos fue construido por alguien que, sin tener ningún conocimiento de robótica ni de electrónica, consiguió poner la máquina en funcionamiento, si bien hubiera sido mejor que Fischertechnik no hubiera confiado en el sentido común de la gente y hubiera proporcionado simplemente un mejor manual de construcción.

Una vez completado el robot, la siguiente tarea consiste en conectarlo a la interface. Aquí, felizmente, el procedimiento es mucho más simple. La placa de interface se compone de varios conectores divididos por secciones que retienen los cables del cable plano. La interface que vemos en la fotografía es para el BBC Micro, y las placas de interface para otros ordenadores probablemente no ofrezcan diferencias sustanciales. En el rincón derecho de la placa hay entradas de interruptores. La lógica de las entradas de interruptores está gobernada por el registro de datos y el registro de dirección de datos del BBC Micro y es similar en operación a la caja buffer que diseñamos en nuestro proyecto del apar-

Electroimán
La barra metálica de encima del electroimán se coloca en el efector final de un brazo-robot. El imán se puede activar y desactivar permitiendo que el robot levante las pequeñas placas metálicas incluidas en el *kit*

Potenciómetro
Se utiliza para realimentar con datos al ordenador, de manera que sea posible evaluar la posición correcta del robot

Motor
Se halla contenido en una carcasa plástica. La conexión al motor se realiza mediante dos conectores. El tornillo giratorio se utiliza para activar mecanismos de tornillo que activan los robots

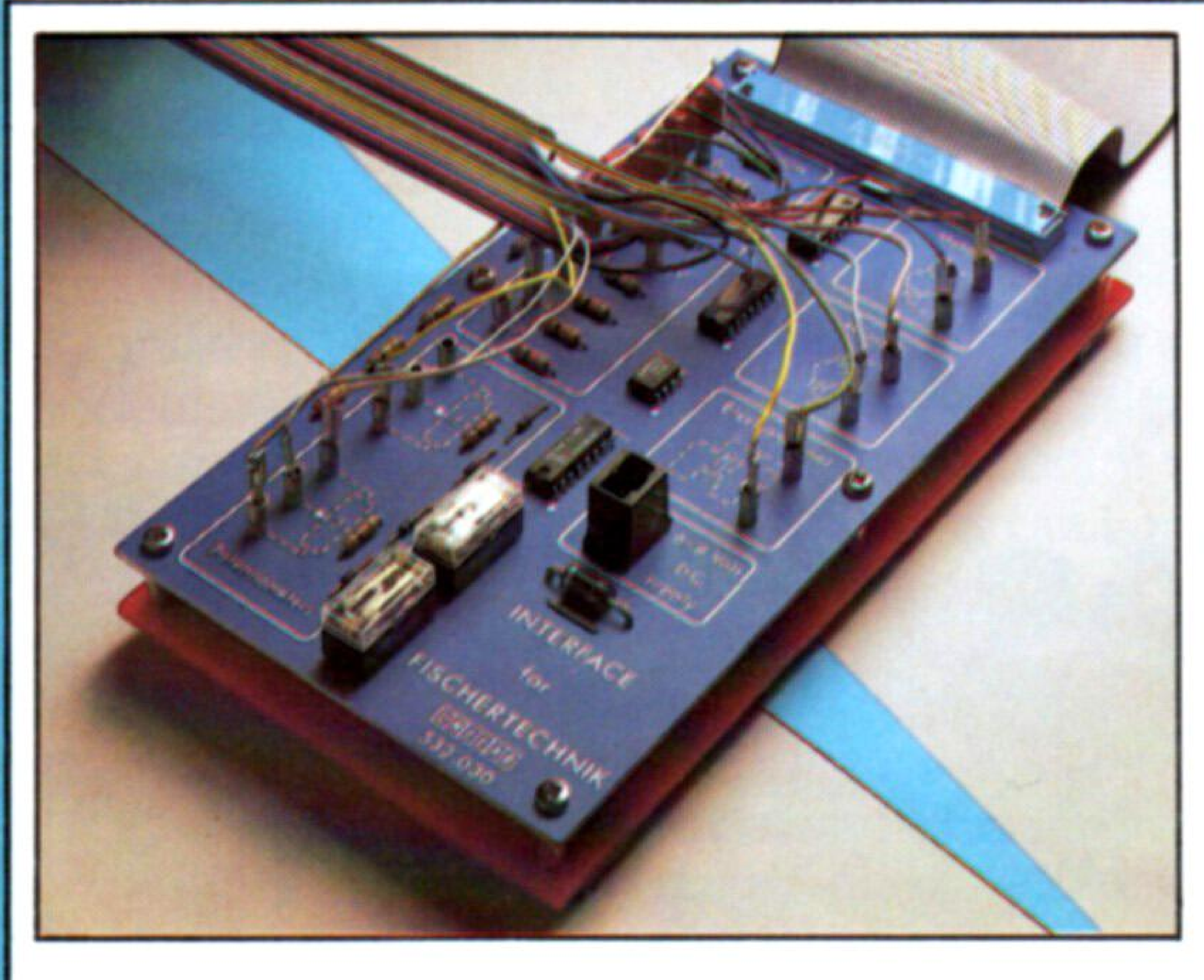
Luces
Siempre que se halle comprometido alguno de los tres sistemas principales de control (imán, motor horizontal o motor vertical), se encenderá una de estas bombillas para informar al usuario cuál es el control que se está utilizando



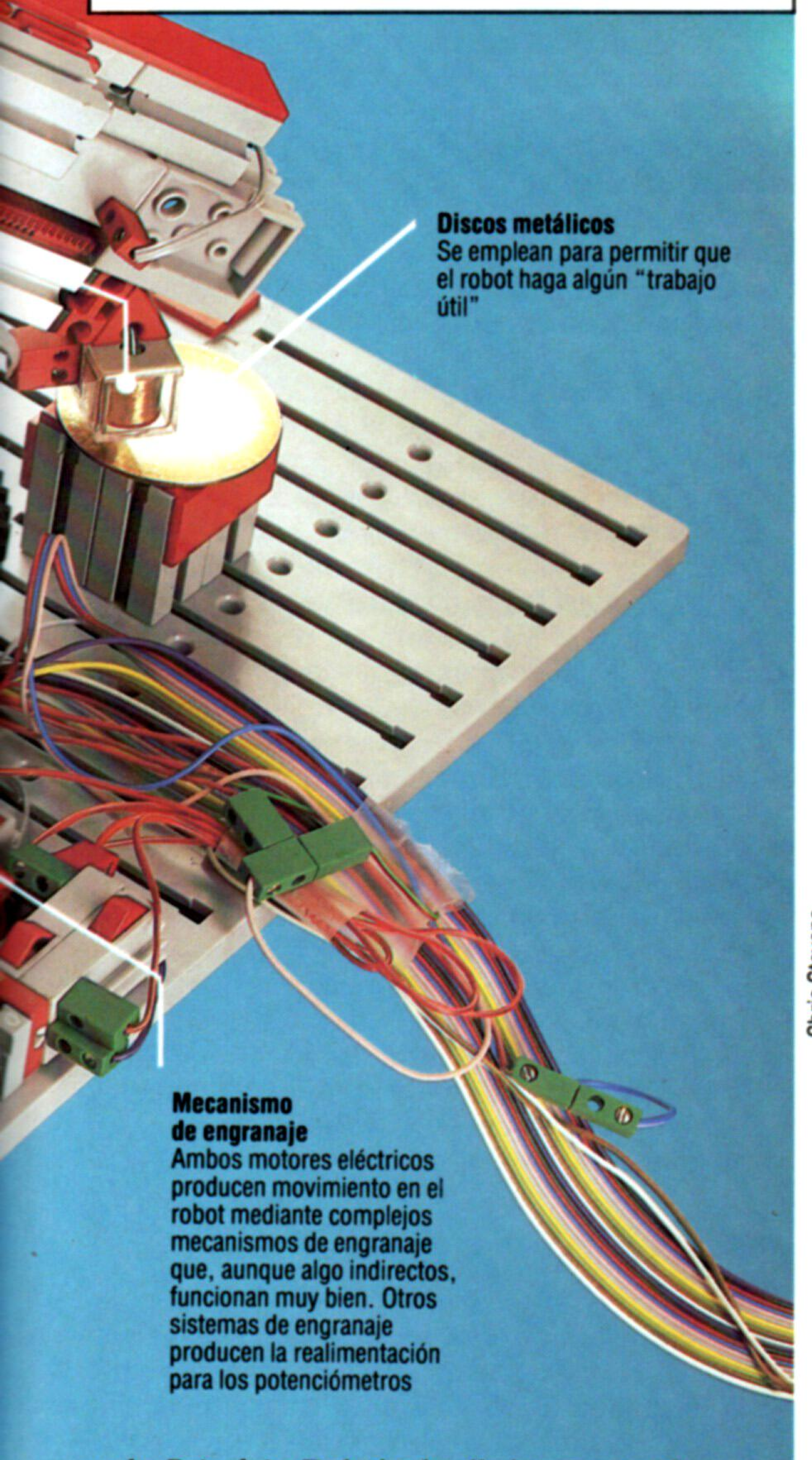
Interruptor
Al pulsar una combinación de los ocho interruptores se genera un número en el registro de la puerta para el usuario, mediante el cual se puede utilizar el registro de dirección de datos para controlar al robot directamente

Ian McKinnell





Potencia para el robot
Para poder controlar el *kit* desde el BBC Micro es necesario adquirir la interface Unilab. Los cables del robot se conectan a la placa de interface, que se une después al micro a través de las puertas para el usuario, analógica y para impresora



Discos metálicos
Se emplean para permitir que el robot haga algún "trabajo útil"

Mecanismo de engranaje
Ambos motores eléctricos producen movimiento en el robot mediante complejos mecanismos de engranaje que, aunque algo indirectos, funcionan muy bien. Otros sistemas de engranaje producen la realimentación para los potenciómetros

Chris Stevens

tado *Bricolaje*. Debajo de ella hay un par de potenciómetros, que permiten que el ordenador controle la potencia que está llegando a los interruptores y las bombillas. El control de los potenciómetros se mide a través de la puerta analógica.

En el otro lado de la placa de interface hay conexiones para controlar los motores y el electroimán (este último lo utiliza el brazo-robot para asir objetos metálicos). Por último, hay un conector para fuente de alimentación de entre 6 y 8 V que alimenta todo el sistema. La interface se conecta al BBC Micro a través de tres puertas separadas (la puerta para el usuario, la puerta analógica y la puerta para impresora) que se emplean para controlar los motores y el electroimán. Debido a que el sistema está diseñado para usuarios sin experiencia, la placa de interface se ha construido para impedir la posibilidad de un cableado incorrecto que pudiera permitir que se les aplicaran a las interfaces del ordenador voltajes que dañaran la máquina.

El manual que se proporciona para la interface (escrito por Unilab, el fabricante del producto) es muy superior al manual de Fischertechnik Robotics. Hay una detallada explicación sobre cómo preparar la interface y cómo ejecutar el software que se suministra con el sistema. Asimismo, hay un breve resumen acerca de las direcciones de control utilizadas para controlar el sistema, permitiendo, por consiguiente, controlar los robots desde BASIC. Sin embargo, el diagrama del cableado parece estar en desacuerdo con el circuito Fischertechnik. Aunque con la interface se proporciona una nota que hace mención a los cables planos diferentes, ésta no parece esclarecer todos los problemas y, nuevamente, uno queda librado a su talento personal para cablear el circuito en la forma correcta.

El lenguaje PROF

Con la interface se suministra una cassette que contiene tres programas para ayudar a controlar los robots. Los dos primeros son programas especializados que controlan el brazo y el plotter, permitiendo controlar dispositivos externos ya sea desde el teclado o bien desde los interruptores de control del propio robot. El tercer programa es PROF, un sistema operativo generalizado a través del cual uno puede dirigir directamente los robots. Esencialmente, es una ampliación de BASIC con la adición de unas pocas instrucciones. Muchas de las palabras extras introducidas en PROF corresponden a nombres de variables tales como Motor y Magnet para la dirección de control, mientras que otras se incluyen para ayudar a editar secuencias de acciones a ejecutar por los robots.

Aunque el PROF es bastante amplio y admite bucles condicionados y programación estructurada, se ciñe mayormente al control de salida. Por consiguiente, el panel para gráficos, un dispositivo de entrada, no está soportado por el PROF, porque el lenguaje no hace ninguna provisión para el almacenamiento de información desde dispositivos de entrada. Ello requiere que uno tenga que escribir sus programas en BASIC.

A pesar de los problemas de la documentación, el *kit* Robotics y la interface constituyen, en conjunto, un sistema muy valioso. Los robots trabajan sorprendentemente bien y se puede construir un modelo operativo desde cero en unas pocas horas. Y el usuario no se ve limitado a tan sólo los seis diseños que se proporcionan. Una vez alcanzada cierta familiaridad con el sistema, la cantidad de proyectos que se pueden construir es inmensa. Por su precio no se trata, ciertamente, de un juguete que pueda adquirirse con calderilla, y, una vez que se supriman los errores existentes en la documentación, el juego para armar Fischertechniks Robotics y la interface serán una introducción ideal para quien tenga un interés general por el campo de la robótica.

FISCHERTECHNIK ROBOTICS KIT

MOVIMIENTO
La potencia de los robots proviene de uno o dos motores eléctricos CD

CONTROL
Ya sea desde el teclado del ordenador o bien a través de los ocho interruptores del propio robot

FUENTE DE ALIMENTACION
Una sola pila PJ996/4R25 de 6 V

PROVEEDORES
Commotion, Computer Operated Motion, 241 Green Street, Enfield EN3 7SJ, Middlesex, Gran Bretaña

Superar las adversidades

Comenzamos la implementación de seis eventos mayores en nuestro programa de simulación de un viaje comercial al Nuevo Mundo durante el s. XVI

En este capítulo comenzaremos a programar las contingencias mayores. Habrá seis de estos eventos, cada uno de los cuales se puede producir sólo una vez durante la travesía. Se los denomina *mayores* porque sus efectos sobre el viaje son más complejos y de mayor alcance que los producidos por los eventos menores. Al comienzo de cada semana se seleccionará al azar uno de los seis acontecimientos, pero si el evento seleccionado ya se hubiera producido durante el viaje, esa semana no tendrá lugar ningún otro acontecimiento mayor.

La línea 49 DIMensiona una matriz, M(), que se utilizará para señalar si alguna de las seis contingencias mayores se ha producido ya. El programa se envía a la subrutina 6500, que genera un acontecimiento desde el bucle principal del programa mediante un GOSUB a la línea 870. La línea 6508 selecciona un número al azar entre 1 y 10 y, dado que sólo hay seis posibles eventos, hay un 60 % de probabilidades de que se seleccione un acontecimiento; de generarse 7, 8, 9 o 10, no se producirá ninguno mayor. Las probabilidades se pueden alterar, si así se desea, regulando la gama del factor aleatorio de esta línea.

Por ejemplo, al cambiar la línea 6508 por

```
X=INT(RND(1)*8)+1
```

se incrementan las probabilidades de seleccionar un acontecimiento mayor a 6 entre 8, es decir, a un 75 %. No obstante, es necesario comprender que si se selecciona un evento mayor y se establece el indicador correspondiente en la matriz M(), entonces la probabilidad de seleccionar un segundo acontecimiento mayor se reduce a 5 entre 8 (un 62,5 %). Las probabilidades de seleccionar un tercer acontecimiento mayor se reduce aún más, a 4 entre 8, o a un 50 %. Por consiguiente, cuanto más eventos se produzcan, menores son las probabilidades de que se produzca otro acontecimiento hasta entonces no seleccionado.

La subrutina de eventos se selecciona utilizando la sentencia ON X GOSUB. Ésta se emplea de la misma forma en que la sentencia ON X GOTO seleccionaba los eventos menores. Si X, el número seleccionado de forma aleatoria, es 1, enviará el programa al primer número de línea de la lista de subrutinas que siga a la sentencia, la subrutina 6530 (el avistamiento de un bote salvavidas). Si X es 2, lo enviará a la segunda dirección de la lista, la subrutina 6700 (la enfermedad mortal). En este capítulo vamos a cubrir solamente las dos primeras contingencias, pero más adelante habremos de añadir a la línea 6510 las direcciones de las cuatro subrutinas restantes.

En la primera subrutina se avista mediante el catalejo un bote salvavidas. En el bote hay cuatro personas y un gran cofre, y el jugador debe decidir si los sube o no a bordo. La desviación añadirá dos días a la duración del viaje y supondrá cuatro bocas más a alimentar, pero si la tripulación se halla por debajo de su fuerza óptima, quizá los brazos extras sean bien recibidos. El programa comprueba primero si el bote salvavidas ya ha sido avistado previamente, examinando el primer elemento de la matriz M() en la línea 6535. Si está establecido en 1, el evento ya se ha producido y retorna al programa principal. Si es 0, la línea 6540 establece M(1) en 1 para impedir que más adelante en el juego el acontecimiento se vuelva a repetir. Como hemos mencionado antes, esta comprobación se utiliza al comienzo de cada una de las contingencias mayores.

Se le pregunta al jugador si se debe recoger a la tripulación del naufragio, en la línea 6576. La línea 6580 comprueba si el primer carácter de la respuesta es S o N, y, si no lo es, retorna a 6568 para volver a formular la pregunta. Si la respuesta es No, el bote salvavidas se aleja flotando y desaparece, y el control retorna al programa principal. Si prevalece la compasión por los náufragos y el jugador decide efectuar el rescate, el programa comprueba si hay espacio suficiente, determinando en primer lugar, en la línea 6625, si la cantidad máxima de tripulantes a bordo está completa. De ser así, en el barco no habrá lugar para los náufragos (lo que se indica mediante CN=16) y el control retorna al programa principal. Para ver si hay lugar para alguno de los supervivientes, la línea 6630 cuenta los espacios vacíos que hay en el barco estableciendo el valor de X en 16 menos el número de tripulantes, CN, y el programa decide si hay lugar para todos los supervivientes en la línea 6630, comprobando si X es mayor que 3. De ser así, son rescatados; pero si X es menor que 3, se le dice al jugador que sólo hay lugar para X personas más, quienes son entonces subidas a bordo.

Los supervivientes se incorporan a la matriz de fortaleza/categoría en el bucle entre las líneas 6638 y 6697. El valor de X se establece inicialmente en 0 y se utiliza para llevar un contador de tripulantes extras. Se efectúa una comprobación a través de la matriz TS(,), para ver si la categoría de tripulante es 0, lo que significa un lugar en la matriz para un tripulante adicional. De hallarse un lugar, el valor de X se incrementa en 1, indicando que ha sido añadido uno de los supervivientes. Cuando X llega a 4, se habrán incorporado todos los supervivientes a la matriz de la tripulación y, por tanto, la línea 6655 establece el valor de T en 16, haciendo que el programa salga del bucle al llegar al NEXT. Si hay espacio para menos de cuatro tripulantes extras, el programa saldrá del bucle cuando la matriz se haya

verificado 16 veces. Los tripulantes extras también se suman al valor de la variable CN, el contador de la tripulación. A cada nuevo tripulante se le asigna un valor al azar para categoría y fortaleza dentro de la matriz de fortaleza/categoría, TS(,); la categoría se selecciona en la escala del 1 al 5 y la fortaleza en la del 50 al 99. Esto significa que la tripulación extra se catalogará como sana o muy sana.

En el bote salvavidas hay, asimismo, un cofre que contiene algunas provisiones. Dado que las designaciones de categoría de tripulante para los cuatro supervivientes del bote salvavidas se seleccionan al azar, esto significa que, por ejemplo, un barco en el que previamente no hubiera ningún médico, podría contar con los servicios de uno gracias a la tripulación del bote salvavidas. En la línea 6685 se prepara un bucle para cada uno de los cuatro tipos de provisiones, y la línea 6690 genera un número al azar entre 10 y 19, que representa la cantidad de cada provisión. En la línea 6692 se imprimen la cantidad, unidades (retenidas en U$(), representando kilos o barriles) y el tipo de provisión. Si durante la semana actual se hubiera agotado alguna provisión, la línea 6693 restablece la cantidad de −999 a 0 para permitir el registro de la adición, y la línea 6694 suma a las existencias actuales las nuevas provisiones.

El segundo evento mayor, la declaración de una epidemia, es llamado mediante el segundo número de línea de la sentencia ON X GOSUB, la subrutina 6700. El impacto de la enfermedad viene determinado por dos factores: que haya o no un médico a bordo, y que durante la etapa de aprovisionamiento se hayan adquirido o no medicinas. El programa comprueba si esta subrutina ya se ha utilizado, de la misma forma que en la contingencia del bote salvavidas, es decir, examinando el valor del segundo elemento de la matriz de indicadores M(), M(2), retornando si M(2)=1.

El programa comprueba luego si hay un médico a bordo. Se prepara un bucle de 1 a 16 para revisar la matriz de fortaleza/categoría de la tripulación en busca de un valor de 2 en la categoría de tripulante, lo que significaría un médico. Si el médico no ha fallecido y su fortaleza no es 0 ni −999, entonces X se establece en 0. Si tras haber finalizado el bucle X conserva el valor 1, ello indica que no hay médico a bordo. Tras comprobar la presencia de un médico, el programa verifica entonces que haya alguna medicina en el barco examinando el valor de PA(1), el elemento que representa las medicinas en la matriz de provisiones. Si hay medicinas a bordo, Y se establece en 0. Si tras esta verificación Y=1, entonces no se dispone de medicinas.

En la línea 6730 se crea un factor de enfermedad, Z. Éste se utiliza para el cálculo de en cuánto se reduce la fortaleza de la tripulación mediante la fórmula Z=((X+Y)*10)+5). Los valores de X e Y dependen de la presencia de un médico y medicinas, de modo que si éstos se encuentran a bordo, la fortaleza se reducirá en 5; de no haber médico o medicinas (una de las dos cosas), la fortaleza se reduce en 15, y si no hubiera ni médico ni medicinas, la fortaleza se reduciría en 25. Se informa al jugador de los motivos por los cuales los efectos de la enfermedad son tan severos mediante las líneas 6734 y 6736, que comprueban la presencia de un médico y medicinas utilizando los valores de X e Y establecidos anteriormente.

Si alguno de los tripulantes está muy débil, es muy probable que la enfermedad lo mate. X se establece en 0 y se emplea para contar los fallecidos. Un bucle entre las líneas 6755 y 6775 recorre la matriz de fortaleza/categoría de la tripulación y reduce las tasas de fortaleza de los tripulantes afectados por la enfermedad. No obstante, ésta no afecta a



todos. La línea 6756 genera un número al azar que, si es menor que 0.3, significa que la fortaleza del tripulante del elemento en curso de la matriz no se ve afectada por la enfermedad. Ello le otorga a cada tripulante alrededor de un 30 % de probabilidades de escapar de la enfermedad (tal vez usted quiera alterar este factor, o, posiblemente, determinarlo al azar). La línea siguiente comprueba si la persona ya está muerta y, por tanto, fuera del alcance de los efectos de la enfermedad. Si el tripulante está vivo, la fortaleza se reduce en Z unidades y se efectúa una comprobación para ver si el tripulante ha muerto a causa de la enfermedad, reduciendo la tasa de fortaleza por debajo de 0. De ser así, se restablece a 999, donde se lo recogerá en el informe de final de semana y el valor de X, la cantidad de fallecidos a causa de la enfermedad, se incrementará en 1. De haber habido medicinas disponibles, lo que se indica mediante Y=0, la línea 6776 reduce la cantidad a la mitad y redondea al frasco entero más próximo. Si durante la epidemia no ha muerto ningún miembro de la tripulación (X=0), entonces el control retorna al programa principal. De lo contrario, se le informa al jugador sobre el número de tripulantes que se vieron mortalmente afectados.

Mód. 8: Dos eventos mayores

Dimensionar matriz bandera

```
49 DIM M(6):REM SEÑALA SI YA SE HAN PRODUCIDO LOS
   EVENTOS MAYORES
```

Adición al bucle principal del viaje

```
870 GOSUB 6500:REM IR A EVENTO MAYOR
```

Rutina de selección de eventos mayores

```
6500 REM EVENTOS MAYORES
6508 X=INT(RND(1)*10)+1
6510 ON X GOSUB 6530,6700
6520 RETURN
```

Rutinas de eventos mayores

```
6530 REM BOTE SALVAVIDAS
6535 IF M(1)=1 THEN RETURN
6536 PRINTCHR$(147)
6540 M(1)=1
6550 S$="SE HA AVISTADO UN BOTE SALVAVIDAS*":GOSUB
     9100
6552 S$="NAVEGANDO A LA DERIVA A LO LEJOS*":GOSUB 9100
6554 PRINT:GOSUB 9200
6556 S$="A TRAVES DEL CATALEJO VES*":GOSUB 9100
6558 S$="QUE CONTIENE:*":GOSUB 9100
6560 PRINT:GOSUB 9200
6562 S$="4 PERSONAS*":GOSUB 9100
6563 GOSUB 9200
6564 S$="Y UN GRAN COFRE!*":GOSUB 9100
6566 PRINT:GOSUB 9200
6568 S$="SI ALTERAS TU RECORRIDO*":GOSUB 9100
6570 S$="PARA RECOGERLOS*":GOSUB 9100
6572 S$="TARDARAS DOS DIAS MAS*":GOSUB 9100
6574 PRINT:GOSUB 9200
6576 S$="QUIERES RESCATARLOS (S/N)*":GOSUB 9100
6578 INPUT I$: I$=LEFT$(I$,1)
6580 IF I$<>"S" AND I$<>"N" THEN 6578
6585 IF I$="S" THEN 6600
6588 PRINT:GOSUB 9200
6590 S$="EL BOTE SALVAVIDAS DESAPARECE...*":GOSUB 9100
6592 PRINT:GOSUB 9200
6594 S$=K$:GOSUB 9100
6596 GET I$:IF I$=""THEN 6596
6599 RETURN
6600 PRINT:GOSUB 9200
6610 EW=EW+2/7
6625 IF CN<>16 THEN 6630
6627 S$="NO PUEDES RECOGERLOS*":GOSUB 9100
6628 S$="PORQUE NO TIENES LUGAR EN EL BARCO*":GOSUB
     9100
6629 GOTO 6592
6630 X=16 - CN:IF X>3 THEN 6635
6632 S$="EN EL BARCO SOLO TIENES LUGAR PARA*":GOSUB
     9100
6633 PRINT X;"PERSONAS MAS"
6634 PRINT:GOSUB 9200
6635 S$="RECOGES:*":GOSUB 9100
6638 X=0
6640 FOR T=1 TO 16
6645 IF TS(T,1)<>0 THEN 6679
6650 X=X+1
6655 IF X>4 THEN T=16:GOTO 6679
6660 CN=CN+1
6665 TS(T,1)=INT(RND(1)*5)+1
6668 TS(T,2)=INT(RND(1)*50)+50
6670 PRINT "1  ";C$(TS(T,1))
6679 NEXT
6680 PRINT:GOSUB 9200
6682 S$="EL COFRE CONTIENE:*":GOSUB 9100
6685 FOR T=1 TO 4
6690 X=INT(RND(1)*10)+10
6692 PRINT X;U$(T);"S DE  ";P$(T)
6693 IF PA(T)= -999 THEN PA(T)=0
6694 PA(T)=PA(T)+X
6695 NEXT
6699 GOTO 6592
6700 REM AZOTE DE LA EPIDEMIA
6705 IF M(2)=1 THEN RETURN
6706 PRINTCHR$(147)
6710 M(2)=1
6712 S$="SE DECLARA UNA EPIDEMIA*":GOSUB 9100
6714 PRINT:GOSUB 9200
6716 X=1
6718 FOR T=1 TO 16
6720 IF TS(T,1)=2 AND TS(T,2)<>0 AND TS(T,2)<>-999THEN
     X=0:T=16
6722 NEXT
6724 Y=1
6726 IF OA(1)<>0 AND OA(1)<>-999 THEN Y=0
6730 Z=((X+Y)*10)+5
6732 I$="TIENES  ":IF X=0 AND Y=0 THEN 6740
6734 IF X=1 THEN S$="SIN NINGUN MEDICO*":GOSUB
     9100:I$="NI  "
6736 IF Y=1 THEN S$=I$+"NINGUNA MEDICINA*":GOSUB 9100
6740 S$="MUCHOS DE LOS TRIPULANTES ESTAN
     AFECTADOS*":GOSUB 9100
6745 PRINT:GOSUB 9200
6750 X=0
6755 FOR T=1 TO 16
6756 IF RND(1)< .3 THEN 6775
6760 IF TS(T,2)=0 OR TS(T,2)=-999 THEN 6775
6765 TS(T,2)=TS(T,2)-Z
6770 IF TS(T,2)<1 THEN TS(T,2)=-999:X=X+1
6775 NEXT
6776 IF Y=1 THEN 6780
6777 S$="SE HAN UTILIZADO LA MITAD DE TUS
     MEDICINAS*":GOSUB 9100
6778 OA(1)=INT((OA(1)/2)+.5)
6780 PRINT:GOSUB 9200
6785 IF X=0 THEN 6797
6790 PRINT"Y";X;
6792 S$="TRIPULANTES MURIERON*"
6794 IF X=1 THEN S$="TRIPULANTE MURIO*"
6795 GOSUB 9100
6796 PRINT:GOSUB 9200
6797 S$=K$:GOSUB 9100
6798 GET I$:IF I$="" THEN 6798
6799 RETURN
```

Complementos al BASIC

Spectrum:

Introduzca las siguientes modificaciones:

```
6510 IF X=1 THEN GO SUB 6530
6511 IF X=2 THEN GO SUB 6700
6536 CLS
6578 INPUT I$:LET I$=I$(1 TO 1)
6596 LET I$=INKEY$:IF I$="" THEN GO TO 6596
6706 CLS
6798 LET I$=INKEY$:IF I$="" THEN GO TO 6798
```

BBC Micro

Introduzca las siguientes modificaciones:

```
6536 CLS
6596 I$=GET$
6706 CLS
6798 I$=GET$
```



Por caminos diferentes

En esta ocasión crearemos el software necesario para controlar nuestro brazo-robot desde un Commodore 64, que será distinto del que se precise para hacerlo desde un BBC Micro

El sistema operativo del Commodore 64 genera interrupciones periódicamente para realizar operaciones de administración doméstica, como puede ser la exploración del teclado. Estas interrupciones se producen regularmente a intervalos de un sesentavo de segundo. El programa en código máquina Commodore intercepta el vector de interrupción para llevar a cabo primero su propia rutina antes de ceder el control a la rutina de interrupciones normal. Este tipo de programa suele denominarse *cuña*.

La primera sección del código máquina se ocupa, por lo tanto, de intercambiar el vector IRQ normal con el vector para nuestra rutina.

Una vez que nuestro vector ha reemplazado al vector IRQ normal, nuestro manejador de eventos se implementará cada vez que se produzca una interrupción IRQ. (El código manejador de eventos es, en gran parte, el mismo que el programa de control para servomotores múltiples que hemos ofrecido anteriormente.) En esencia, este código envía impulsos de corriente a lo largo de las líneas de datos desde la puerta para el usuario hacia los motores. La duración de cada impulso le informa al motor correspondiente el ángulo que debe adoptar. La rutina en código máquina recibe la información que determina la longitud de los impulsos del motor desde una tabla de ocho bytes, denominada ANGULO. Al igual que en la versión para el BBC Micro, se utiliza un trozo de código adicional para cargar las posiciones ANGULO uniformemente desde una segunda tabla, NUEVAPOS, que es la tabla que se carga desde el programa controlador en BASIC.

El programador de secuencias para el brazo utiliza el programa en código máquina que acabamos de describir para activar el brazo-robot. Similar a la versión para el BBC Micro, este programa permite controlar el brazo desde el teclado, guardando las posiciones clave, que se pueden luego reproducir o almacenar en cinta o disco. Entre esta versión y la versión para el BBC Micro existen, sin embargo, una o dos diferencias importantes. En primer lugar, al controlar el brazo-robot los motores tienen una tendencia a oscilar a través de una posición dada. Ello se debe a que el chip de video VIC-II del Commodore interfiere con la generación regular de interrupciones. La forma más simple de evitar este problema consiste en limpiar la pantalla cuando se emplea el programa cuña. Las subrutinas de las líneas 7000 y 8000 limpian la pantalla antes de entrar en la cuña y la restablecen al salir de la misma.

El segundo problema es que el BASIC Commodore hace que la comprobación de una gran cantidad de teclas (y las bifurcaciones correspondientes) sea un proceso lento. Por consiguiente, se utiliza la última subrutina en código máquina para comparar el código ASCII de una pulsación de tecla con una tabla de posibles teclas que tienen una función en la tabla. Cuando se encuentra una coincidencia en la tabla, se pasa como un número que puede utilizar el ON ... GOSUB del BASIC para llamar a la rutina adecuada.

Programador de secuencias para el brazo en el C64

Cargador en BASIC

```
10 REM ******************************
20 REM **                          **
30 REM **   CARGADOR DE BASIC PARA  **
40 REM **   CONTROLADOR BRAZO CBM   **
50 REM **                          **
60 REM ******************************
65 :
70 FOR I=49155 TO 49394
75 READ A:POKE I,A:CC=CC+A
80 NEXT I
90 READ CS:IF CS<> CC THEN PRINT"ERROR EN SUMA DE
   CONTROL":STOP
95 :
100 DATA169,255,141,3,221,169,62,141,1
110 DATA192,169,192,141,2,192,120,173
120 DATA20,3,174,1,192,141,1,192,142
130 DATA20,3,173,21,3,174,2,192,141,2
140 DATA192,142,21,3,169,16,133,251
150 DATA169,193,133,252,169,255,160,0
160 DATA145,251,136,208,251,88,96,8,72
170 DATA152,72,138,72,169,255,141,1
180 DATA221,162,7,169,255,24,106,72
190 DATA188,0,193,49,251,145,251,104
200 DATA202,16,243,160,48,136,208,253
210 DATA169,255,160,0,49,251,141,1,221
220 DATA200,208,248,162,7,169,255,188
230 DATA0,193,145,251,202,16,248,104
240 DATA170,104,168,104,40,108,1,192
250 DATA162,0,160,0,189,8,193,221,0
260 DATA193,240,14,176,6,222,0,193,76
270 DATA156,192,254,0,193,76,156,192
280 DATA200,232,224,4,208,228,32,169
290 DATA192,192,4,208,217,96,138,72
300 DATA152,72,160,255,174,0,192,136
310 DATA234,234,208,251,202,208,248
320 DATA104,168,104,170,96,0,255,255,0
330 DATA0,255,255,0,0,255,255,0,0,255
340 DATA255,0,0,72,138,72,152,72,32
350 DATA228,255,201,0,240,249,162,0
360 DATA221,191,192,240,7,232,224,17
370 DATA208,246,162,255,142,207,192
380 DATA104,168,104,170,104,96
390 DATA33097:REM*SUMA DE CONTROL*
```

Listado assembly

```
;..........................................
;..........................................
;..                                      ..
;..   CONTROLADOR BRAZO CBM              ..
;..                                      ..
;..........................................
;..........................................
;
PUERTA=56577        ;REGISTRO DATOS PUERTA USUARIO
DDR=56579           ;REG DIRECCION DATOS PUERTA USUARIO
ANGULO=$C100        ;POSICION VALOR ANGULO
NUEVAPOS=$C108      ;TABLA POSICION
MOTTAB=$C110        ;TABLA REFERENCIA MOTOR
ZPAGE=$FB           ;PUNTERO PAGINA 0 A TABLA
IRQVEC=$0314        ;LOBYTE VECTOR IRQ
;
*=$C000
DELFAC *=*+1;ALMACENAMIENTO PARA FACTOR DEMORA
OURVEC *=*+2;ALMACENAMIENTO PARA NUESTRO VECTOR
        LDA  #$FF
        STA  DDR            ;ESTABLECER DDR EN SALIDA
        LDA  #<EVENT
        STA  OURVEC         ;APUNTA A EVENT
        LDA  #>EVENT        ;MANEJADOR
        STA  OURVEC+1
;
        SEI                 ;APAGAR INTERRUPCIONES
        LDA  IRQVEC         ;INTERCAMBIAR VECTOR
        LDX  OURVEC         ;IRQ EXISTENTE POR
        STA  OURVEC         ;VECTOR NUESTRO
        STX  IRQVEC
        LDA  IRQVEC+1
        LDX  OURVEC+1
        STA  OURVEC+1
        STX  IRQVEC+1
;
;.... INICIALIZAR TABLA ....
;
        LDA  #<MOTTAB
        STA  ZPAGE
        LDA  #> MOTTAB
        STA  ZPAGE+1
;
        LDA  #$FF
        LDY  #$00
TABLA
        STA  (ZPAGE),Y
        DEY
        BNE  TABLA
        CLI                 ;ENCENDER INTERRUPCIONES
;
;.... MANEJADOR EVENTOS ....
;
EVENT
        PHP
        PHA                 ;GUARDAR REGISTROS
        TYA                 ;EN PILA
        PHA
        TXA
        PHA
;
;..EMPEZAR IMPULSO, PARA ALGUNOS MOTORES SE
;PODRIA EMPEZAR ANTES DE RELLENAR LA TABLA
;Y ASI REDUCIR EL BUCLE DE ESPERA DE ABAJO..
;
        LDA  #$FF
        STA  PUERTA
;.. LLENAR TABLA CON EXCEPCIONES ..
        LDX  #$07
        LDA  #$FF
        CLC
EXCEPT
        ROR  A
        PHA                 ;PATRON BITS
        LDY  ANGULO,X       ;TOMAR DESPL. MOTOR X
        AND  (ZPAGE),Y      ;CONSERVAR PATRON EXISTENTE
        STA  (ZPAGE),Y      ;PERO MODIFICADO PARA MOTOR X
        PLA
        DEX
        BPL  EXCEPT
;.. AHORA LA TABLA ESTA CARGADA ..
        LDY  #$30
WAIT
        DEY                 ;ESPERAR
        BNE  WAIT           ;UN POCO
;
        LDA  #$FF           ;TODOS IMPULSOS ENCENDIDOS
        LDY  #$00
LOOP
        AND  (ZPAGE),Y      ;PERO DESENMASCARA CADA ELEMENTO
        STA  PUERTA         ;DE LA TABLA POR TURNO
        INY
        BNE  LOOP
;
        LDX  #$07
        LDA  #$FF
CLEAR
        LDY  ANGULO,X       ;BORRAR TODAS EXCEPCIONES
        STA  (ZPAGE),Y
        DEX
        BPL  CLEAR
;.. TODOS LOS IMPULSOS DEBEN HABER TERMINADO AHORA ..
        PLA
        TAX
        PLA                 ;RESTAURAR REGISTROS
        TAY
        PLA
        PLP
        JMP  (OURVEC)       ;SALTAR A RUTINA
                            ;IRQ NORMAL
;
;.... MOVER EL MOTOR UNIFORMEMENTE ....
;
LOOP2
        LDX  #$00
        LDY  #$00
LOOP1
        LDA  NUEVAPOS,X;NUEVA POSICION PARA SERVO X
        CMP  ANGULO,X       ;ES LA MISMA QUE LA POS. ANTIGUA?
        BEQ  MOVED          ;SI=NO HACER NADA
        BCS  ADD            ;SI>SUMAR
        DEC  ANGULO,X       ;SINO RESTAR
        JMP  INCRX
ADD
        INC  ANGULO,X
        JMP  INCRX
MOVED
        INY
INCRX
        INX
        CPX  #$04           ;HECHOS LOS 4 SERVOS
        BNE  LOOP1          ;SI NO SIGUIENTE SERVO
        JSR  WAITER         ;BREVE PAUSA
        CPY  #$04           ;SI Y=4 ENTONCES TODOS MOVIDOS
        BNE  LOOP2
        RTS
;
WAITER
        TXA
        PHA                 ;GUARDAR REGS X & Y
        TYA
        PHA
        LDY  #$FF
        LDX  DELFAC         ;TOMAR FACTOR DEMORA
ENCORE
        DEY
        NOP                 ;256*8 CICLOS RELOJ
        NOP                 ;ESPERA
        BNE  ENCORE
        DEX                 ;SI MAS DEMORA ENTONCES
        BNE  ENCORE         ;REPETIR
        PLA
        TAY
        PLA
        TAX
        RTS
;... COMPROBAR TECLADO ....
;
KEYTAB *=*+16
RESULTADO *=*+1
GETIN=$FFE4
;
CHECK
        PHA
        TXA
        PHA
        TYA
        PHA
NOKEY
        JSR  GETIN          ;TOMAR UN CARACTER
        CMP  #$00           ;NINGUN CAR?
        BEQ  NOKEY
        LDX  #$00
COMPAR
        CMP  KEYTAB,X
        BEQ  EXIT           ;ESTA EL CAR. EN LA TABLA?
        INX
        CPX  #17
        BNE  COMPAR
        LDX  #$FF           ;NO HALLADA SEÑAL
EXIT
        STX  RESULTADO      ;ALMACENAR DESPL. KEYTAB
        PLA
        TAY
        PLA
```

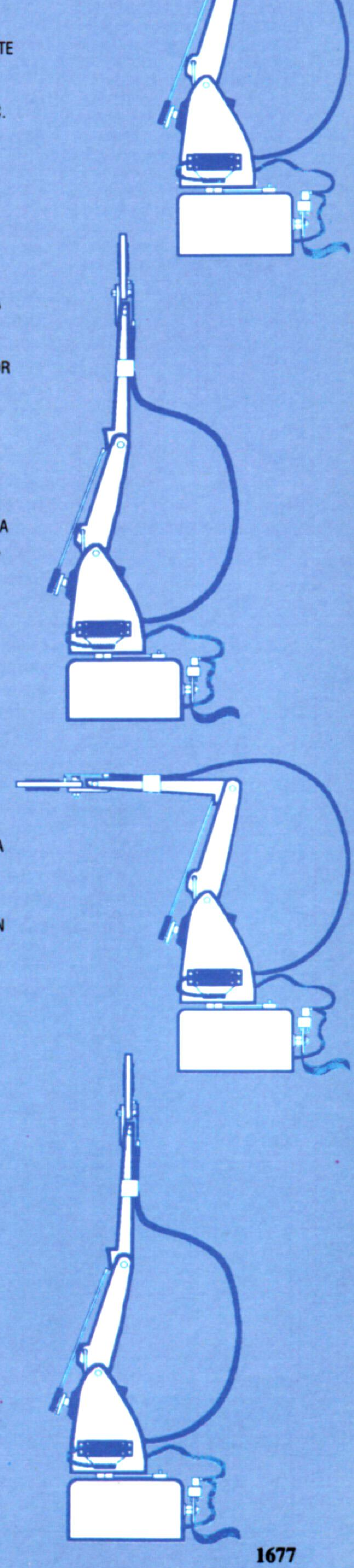
Ian McKinnell



```
TAX
PLA
RTS
```

Programador de secuencias para el brazo-robot

```
10 REM  **********************
20 REM * *********************
30 REM **                   **
40 REM **  PROGRAMADOR CBM  **
50 REM **  SECUENCIAS BRAZO **
70 REM **                   **
80 REM  **********************
90 REM  **********************
95 :
96 DN=8:REM PARA CASSETTE DN=1
97 IF A=0 THEN A=1:LOAD"ROBARM.HEX",DN,1
100 GOSUB 1000:REM PREPARACION
110 PRINT CL$
120 X=5:Y=6:GOSUB 10000:PRINT"TE GUSTARIA:  "
130 X=4:Y=10:GOSUB 10000:PRINT"1....PROGRAMAR NUEVA
    SECUENCIA BRAZO"
140 X=4:Y=12:GOSUB 10000:PRINT"2....AGREGAR
    MOVIMIENTOS A UN PROGRAMA"
150 X=4:Y=14:GOSUB 10000:PRINT"3....REPRODUCIR UN
    ARCHIVO"
160 X=4:Y=16:GOSUB 10000:PRINT"4....SALIR DEL
    PROGRAMA"
170 GET G$:IF G$="" THEN 170
180 IF G$="1" THEN C=0:LM=0:REM RESTABLECER PUNTEROS
    MATRIZ
190 IF G$="1" OR G$="2" THEN GOSUB 2000:GOSUB
    3000:GOSUB 4000:GOSUB 5000
200 IF G$="3" THEN GOSUB 6000:GOSUB 4000
210 IF G$="4" THEN PRINT CL$:END
220 GOTO 110
500 REM **** FUNCIONES ****
540 DX=DX+1:RETURN
550 DX=DX-1:IF DX<0 THEN DX=0
555 RETURN
560 P=PEEK(NP)+DX:IF P<256 THEN POKENP,P
565 RETURN
570 P=PEEK(NP)-DX:IF P>0 THEN POKENP,P
575 RETURN
580 P=PEEK(NP+1)+DX:IF P<256 THEN POKENP+1,P
585 RETURN
590 P=PEEK(NP+1)-DX:IF P>0 THEN POKENP+1,P
595 RETURN
600 P=PEEK(NP+2)-DX:IF P>0 THEN POKENP+2,P
605 RETURN
610 P=PEEK(NP+2)+DX:IF P<256 THEN POKENP+2,P
615 RETURN
620 P=PEEK(NP+3)+3*DX:IF P<256 THEN POKENP+3,P
625 RETURN
630 P=PEEK(NP+3)-3*DX:IF P<256 THEN POKENP+3,P
635 RETURN
640 FOR I=0 TO 3:POKE NP+I,R%(C,I):NEXT I:RETURN
650 C=C+1:IF C>MC THEN C=0
655 RETURN
660 C=C-1:IF C<0 THEN C=MC
665 RETURN
670 FOR I=0 TO 3:R%(C,I)=PEEK(NP+I):NEXT:C=C+1:RETURN
680 FOR I=0 TO 3:R%(C,I)=PEEK(NP+I):NEXT:C=C+1:RETURN
999 :
1000 REM **** PREPARACION ****
1010 C=0:NS=3:REM N.DE SERVOS-1
1020 LM=0:OC=0:DF=10:REM FACTOR DE DEMORA
1030 MC=100:REM MAX CONTADOR
1040 DIM R%(MC,NS):REM MATRIZ POSICIONES TECLAS
1050 CL$=CHR$(147):REM LIMPIAR PANTALLA
1060 DW$="":FOR I=1 TO 25:DW$=DW$+CHR$(17):NEXT I:REM
     CURSOR ABAJO
1065 POKE 650,128:REM EST. TECLAS PARA REPETIR
1070 DL=49152:NP=49416:REM DIRECCIONES DEMORA Y
     POSICION
1075 GS=49155:OFF=49170:REM DIRECCIONES SISTEMA CUÑA
     ON/OFF
1077 MOVER=49281:REM MOVER DIRECCIONES SYS SERVOS
1078 KEYTAB=49343:RESULTADO=49359:REM DIRECCIONES
     PULSACION TECLA
1079 CHECK=49360:REM DIRECCION SYS COMPROBACION TECLA
1080 REM ** LEER DATOS ASC PULSACION TECLA **
1090 FOR I=0 TO 14:READ A:POKE KEYTAB+I,A:NEXT I
1100 DATA 73,68,157,29,145,17,65,90,88,67
1110 DATA 82,78,66,83,81
1200 REM ** PONER A CERO POSICIONES NUEVAPOS **
1210 FOR I=0 TO 3:POKENP+I,0:NEXT I
1990 RETURN
1999 :
2000 REM **** INFORME ****
2010 PRINT CL$
2020 X=5:Y=5:GOSUB 10000:PRINT"POR FAVOR UTILIZA :"
2030 X=1:Y=7:GOSUB 10000:PRINT"TECLAS CURSOR....PARA
     D/I Y 1ER. BRAZO AR/AB"
2040 X=1:Y=9:GOSUB 10000:PRINT"A & Z........... PARA
     2DO.BRAZO AR/AB"
2050 X=1:Y=11:GOSUB 10000:PRINT"X & C........... PARA
     ABRIR/CERRAR PINZA"
2060 X=1:Y=13:GOSUB 10000:PRINT"S............... PARA
     SALVAR UNA POSICION"
2070 X=1:Y=15:GOSUB 10000:PRINT"Q............... PARA
     VOLVER AL MENU"
2080 X=1:Y=17:GOSUB 10000:PRINT"R............... PARA
     MOVER A POSICION SALVADA"
2090 X=1:Y=19:GOSUB 10000:PRINT"N & B........... SIGUIENTE
     Y PREVIO CONTADOR"
2100 X=1:Y=21:GOSUB 10000:PRINT"E...............
     ESTABLECER NUEVO CONTADOR"
2110 X=1:Y=23:GOSUB 10000:PRINT"I & D........... PARA INC.
     Y DEC. VELOCIDAD"
2120 X=2:Y=2:GOSUB 10000:PRINT"CONTADOR=";C;"  ";
2130 FOR I=0 TO 3:PRINT R%(C,I);"  ";:NEXT:PRINT
2135 GET A$:IF A$="" THEN 2135
2140 RETURN
2999 :
3000 REM **** PROGRAMAR BRAZO ****
3005 GOSUB 7000:REM BORRAR PANTALLA Y EMPEZAR CUÑA
3010 POKE DL,1:REM ESTABLECER FACTOR DEMORA EN 1
3020 DX=8:REM CAMBIAR VELOCIDAD
3030 SYS CHECK:REM COMPROBAR PULSACION TECLA
3035 R=PEEK(RESULTADO)+1:IF R>15 THEN 3030:REM TECLA
     NO VALIDA
3037 ON R GOSUB 540,550,560,570,580,590,600,610,620,
     630,640,650,660,670
3190 IF C>LM THEN LM=C-1:REM REGISTRAR MAX CONTADOR
     HASTA AHORA
3220 OC=C
3260 SYS MOVER:REM MOVER SERVOS C/M
3270 IF R=15 OR C>MC THEN GOSUB 8000:RETURN:REM
     APAGAR CUÑA Y SALIR
3280 GOTO 3030:REM REPETIR
3999 :
4000 REM **** REPRODUCIR SECUENCIA MOVIMIENTOS ****
4010 PRINT CL$
4020 X=5:Y=23:GOSUB 10000:PRINT"REPRODUCIR SECUENCIA
     (S/N),R REPITE"
4030 GET AN$:IF AN$<>"S" AND AN$<>"N" AND AN$<>"R"
     THEN 4030
4040 IF AN$="N"THEN RETURN
4050 IF AN$<>"R" THEN X=5:Y=22:GOSUB
     10000:INPUT"FACTOR DE DEMORA 1-255";DF
4060 IF DF<0 OR DF>255 THEN 4050:REM COMPROBAR
     ESCALA
4070 POKE DL,DF:REM ESTABLECER REGISTRO FACTOR
     DEMORA
4075 GOSUB 7000:REM BORRAR PANTALLA Y COMENZAR
     CUÑA
4080 FOR I=0 TO LM
4090 X=5:Y=2:GOSUB 10000:PRINT"N. EN SECUENCIA=  ";I;
     "  ";
4100 FOR S=0 TO 3
4110 POKE NP+S,R%(I,S)
4120 NEXT S:SYS MOVER:REM MOVE SERVOS C/M
4130 NEXT I
4135 GOSUB 8000:REM RESTAURAR PANTALLA Y APAGAR CUÑA
4140 GOTO 4010:REM REPETIR
4999 :
5000 REM **** GUARDAR UNA SECUENCIA ****
5010 PRINT CL$
5020 X=5:Y=10:GOSUB 10000:PRINT"GUARDAR SECUENCIA EN
     ARCHIVO (S/N)"
5030 GET AN$:IF AN$<>"S" AND AN$<>"N" THEN 5030
5040 IF AN$="N" THEN RETURN
5050 X=5:Y=12:GOSUB 10000:INPUT"NOMBRE ARCHIVO";FL$
5060 OPEN 3,DN,3,"Q0:"+FL$+".DATA,S,W":REM ABRIR
     ARCHIVO
5070 PRINT #3,LM
5080 FOR A=0 TO LM:FOR B=0 TO 3
5090 PRINT #3,R%(A,B):NEXT B,A
5100 PRINT #3:CLOSE3
5120 RETURN
5999 :
6000 REM **** CARGAR UN ARCHIVO ****
6010 PRINTCL$
6020 X=5:Y=10:GOSUB 10000:INPUT"NOMBRE ARCHIVO A
     CARGAR";FL$
6040 OPEN 3,DN,3,"Q0:"+FL$+".DATA,S,R":REM ABRIR
     ARCHIVO
6050 INPUT #3,C:LM=C
6060 FOR A=0 TO C:FOR B=0 TO 3
6070 INPUT #3,R%(A,B):NEXT B,A
6080 PRINT #3:CLOSE3
6100 RETURN
6999 :
7000 REM **** COMENZAR CUÑA ****
7010 POKE 53265,PEEK(53265)AND 239:REM BORRAR
     PANTALLA
7015 SYS GS
7020 RETURN
7999 :
8000 REM **** DETENER CUÑA ****
8010 POKE 53265,PEEK(53265)OR 16:REM BORRAR PANTALLA
8015 SYS OFF
8020 RETURN
8999 :
10000 REM **** POSICIONAR CURSOR EN X,Y ****
10010 PRINTCHR$(19);:PRINTTAB(X);LEFT$(DW$,Y);
10020 RETURN
```



Trabaja la vista

Vamos primero a examinar los gráficos a colores y la organización de la memoria que necesita el chip VIC-II, para obtener después inusitados efectos visuales

El Commodore 64 tiene ocho modos básicos de gráficos en pantalla. En los gráficos de baja resolución el carácter puede albergarse en la ROM o en la RAM y visualizarse en uno de estos tres modos: estándar, multicolor o ampliado *(extended)*. Se distinguen además dos modos en alta resolución: estándar y multicolor. Aparte de estos modos básicos, se dispone de diversas variantes: se puede hacer que la pantalla cuente con 38 columnas, en vez de las 40 columnas habituales y/o 24 filas, en vez de 25. Tales características suelen usarse junto con el *scroll* (desplazamiento) lento vertical u horizontal. El *scroll* en el Commodore 64 se controla mejor desde código máquina, ya que los datos a visualizar en desplazamiento deben pasar bastante rápidamente por la RAM de pantalla. Ya diseñamos anteriormente un programa para desplazamientos suaves horizontales de pantalla empleando el modo de pantalla de 38 columnas.

Si una pantalla en alta resolución es empleada con un programa largo, puede que la memoria se resienta. El Commodore 64 facilita una gran libertad en el posicionamiento de la pantalla en memoria tanto en alta como en baja resolución, por lo que comenzaremos con un breve análisis de cómo puede el programador trasladar ambas pantallas por la memoria. Veremos en el siguiente capítulo que si se emplea el código máquina para hacer los puntos, incluso se puede colocar la pantalla en alta resolución *detrás* de la ROM del intérprete del BASIC. Bien pensado, se trata de un excelente recurso dado que si sólo se está ejecutando código máquina la ROM del intérprete ocupa ocho Kbytes de una memoria valiosa inútilmente. Una interesante coincidencia nos dice que son justamente ocho Kbytes los que se necesitan para una pantalla de alta resolución.

Esta función del Chip para Interface de Video 6566/67 (VIC-II) para generar datos de visualización del video es la que se pasa al televisor o pantalla. Para ello el VIC-II debe saber leer datos desde la RAM o la ROM. Merece la pena estudiar el modo como el VIC-II obtiene sus datos, ya que esto es la base del comportamiento más recóndito del Commodore 64.

Modo multicolor

Ya quedaron descritos tanto la visualización habitual en baja resolución como el modo en mapa de bits en el Commodore 64. Vamos a examinar ahora otras dos maneras de emplear los gráficos de la máquina.

Contrastando con dos de ellos, podemos obtener, en el modo multicolor, cuatro colores dentro de la celda individual de un carácter. Pero en esta resolución horizontal perdemos en otro aspecto, y es que ahora ésta trabaja por pares de pixels y no pixel a pixel. El modo multicolor puede emplearse con gráficos tanto en alta como en baja resolución, aunque los puntos coloreados se obtienen de un modo algo distinto en el modo de alta resolución. El siguiente POKE activa y desactiva desde el BASIC dicho modo multicolor.

```
POKE 53270,PEEK (53270)OR16
POKE 53270,PEEK (53270)AND239
```

En baja resolución, si el modo multicolor está activo y el bit 4 del cuarteto o semibyte asociado del color está puesto a uno, el carácter se interpreta en modo multicolor, donde los tres bits inferiores especifican el color. Esto quiere decir que los caracteres que tienen cuartetos asociados de color con valor entre 0 y 7 son interpretados normalmente; mientras que si el código del color está entre 8 y 15, el carácter se visualizará en modo multicolor. En el cuadro 1 mostramos cómo se determinan los colores de cada par de pixels.

Con el contenido de las direcciones 53282 y 53283, podemos cambiar instantáneamente el color de cada par de pixels asociado en Multicolor. Es de resaltar que el Multicolor trabaja mejor con caracteres definidos por el usuario, es decir, las agrupaciones de bits tienen en cuenta el hecho de ser interpretados como pares.

Otro modo de gráficos disponible para un programador del Commodore 64 es el modo de color ampliado *(extended)*. Este modo permite controlar el color de fondo de los primeros 64 caracteres dentro de la matriz de caracteres. El modo de color ampliado no puede emplearse junto con el modo multicolor. Las siguientes líneas de BASIC activan y desactivan el modo de color ampliado respectivamente:

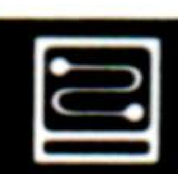
```
POKE 53265,PEEK(53265)OR64
POKE 53265,PEEK(53265)AND191
```

Cuadro 1

Modelo de bits del par de pixels		Color	Determinado por
0	0	Fondo de pantalla	53281 ($D021) Bits 0 al 4
0	1	Multicolor #1	53282 ($D022) Bits 0 al 4
1	0	Multicolor #2	53283 ($D023) Bits 0 al 4
1	1	Color primer plano	Bits 0 al 3 del cuarteto color

Cuadro 2

Código pantalla	Bit 7	Bit 6	Registro color fondo
64 — 127	0	1	53282 ($D022) Color 1 ampliado
128 — 191	1	0	53283 ($D023) Color 2 ampliado
192 — 255	1	1	53284 ($D024) Color 3 ampliado



Un mismo carácter puede aparecer en la pantalla hasta con cuatro colores de fondo diferentes, uno de los cuales es el color de fondo de la pantalla. Los caracteres restantes no pueden visualizarse en el modo de color ampliado, ya que los bits 6 y 7 del código de pantalla se emplean para controlar indirectamente el color del carácter. Por ejemplo, el código de pantalla para "A" es 1, y el de "A" en un campo invertido es 65. Pero si introducimos (POKE) 65 en la pantalla en modo de color ampliado, no obtendremos una "A" en campo invertido sino una "A" habitual con un color de fondo determinado por el contenido de la dirección 53282 ($D022). Igualmente, introduciendo (POKE) 129 en la pantalla obtendremos una "A" normal, pero con un color de fondo dictado por el contenido de la dirección 53283 ($D023). El cuadro 2 muestra la relación que existe entre los códigos de pantalla y los registros de color.

Gestión de la memoria

El chip VIC-II ve desde el 6510 un mapa distinto y mucho más simple de memoria. Simultáneamente el VIC-II sólo puede ver uno de los cuatro bloques de 16 Kbytes (bancos) de memoria. Podemos imaginarnos este banco de 16 Kbytes como la "ventana" del VIC-II sobre la memoria. La dirección de base de esta ventana puede tomar uno de los cuatro valores que se encuentran bajo el control del software

```
1000 REM ** SELECCION BANCO/VENTANA DEL
     VIC **
1010 POKE 56578,PEEK(56578)OR3:REM BITS 0,1
     DE CIA#2 DE RDD,PARA SALIDA
1020 VT=3:REM SELECCIONA LA VENTANA
     NORMAL
1030 POKE 56576,PEEK(56576)AND252)ORVT:REM
     BITS 0,1 DE CIA#1 PUERTA A
```

VT puede tomar aquí valores del 0 al 3. Cada vez encontraremos el valor de VT en curso como PEEK (56576)AND3. La dirección correspondiente a la parte inferior de la ventana de 16 Kbytes sobre la memoria se calcula por medio de la fórmula: VB=16384*(3−VT), en la que el valor VT proporciona las direcciones correspondientes:

VT	Dirección inicio ventana
0	49152 ($C000)
1	32768 ($8000)
2	16384 ($4000)
3	0 ($0000)

Dentro de la ventana del VIC-II, el 6510 espera ver la memoria de pantalla y una imagen de la ROM de caracteres en baja resolución (o datos en alta resolución, si se ha seleccionado el modo de alta resolución). Puede que también necesite encontrar datos de sprites, y en ese caso los ha de ver dentro de la ventana.

Los punteros de cada uno de los ocho sprites se encuentran colocados al final de la memoria de pantalla (y con la pantalla deberán moverse).

El desplazamiento del inicio de la memoria de pantalla desde la base de la actual ventana del VIC-II se controla por medio de los cuatro bits superiores del registro de control del VIC-II en la dirección 53272 ($D018). Empleando estos cuatro bits, podemos colocar la pantalla en cualquiera de los bloques de 16 Kbytes que integran la ventana.

```
1040 REM ** SELECCION DESPLAZAMIENTO
     PANTALLA DE LA BASE DE LA VENTANA **
1050 SD=1:REM SELECCIONA EL
     DESPLAZAMIENTO NORMAL
1060 POKE 53272,(PEEK(53272)AND15)OR
     16*SD
```

SD puede tomar valores entre 0 y 15. Siempre encontraremos el valor en curso de SD empleando PEEK(53272)AND15. La dirección correspondiente a la base de la pantalla actual en la memoria se calcula o bien así PT=VB+1024*SD (base ventana más desplazamiento) o bien así:

```
PT=16384*(3-(PEEK(56576)AND3))+64*(PEEK
(53272)AND240)
```

El problema a la hora de mover la pantalla de un sitio a otro radica en que la RAM del color no se mueve. Por ello, si deseamos obtener una pantalla alternativa, habremos de emplear una pequeña rutina en código máquina para intercambiar la memoria del color dentro o fuera de un buffer o memoria intermedia según convenga. Es lo que se hizo como base de un programa para pantallas alternativas en el Commodore 64.

Otro factor a considerar es que si se desea imprimir (PRINT) en la nueva pantalla, es necesario avisar al *sistema operativo* (y no al chip de video) sobre dónde se encuentra la nueva pantalla. Esto se realiza con un POKE (o STA) en la dirección 648 ($0288), que es un puntero que contiene el byte superior de la dirección de base de la memoria de pantalla. Si calculamos PT como se ha hecho más arriba, este trabajo lo hará la siguiente instrucción en BASIC:

```
POKE 648,INT(PT/256)
```

Para seleccionar la dirección de base de la matriz de caracteres o pantalla en alta resolución, emplearemos el siguiente código:

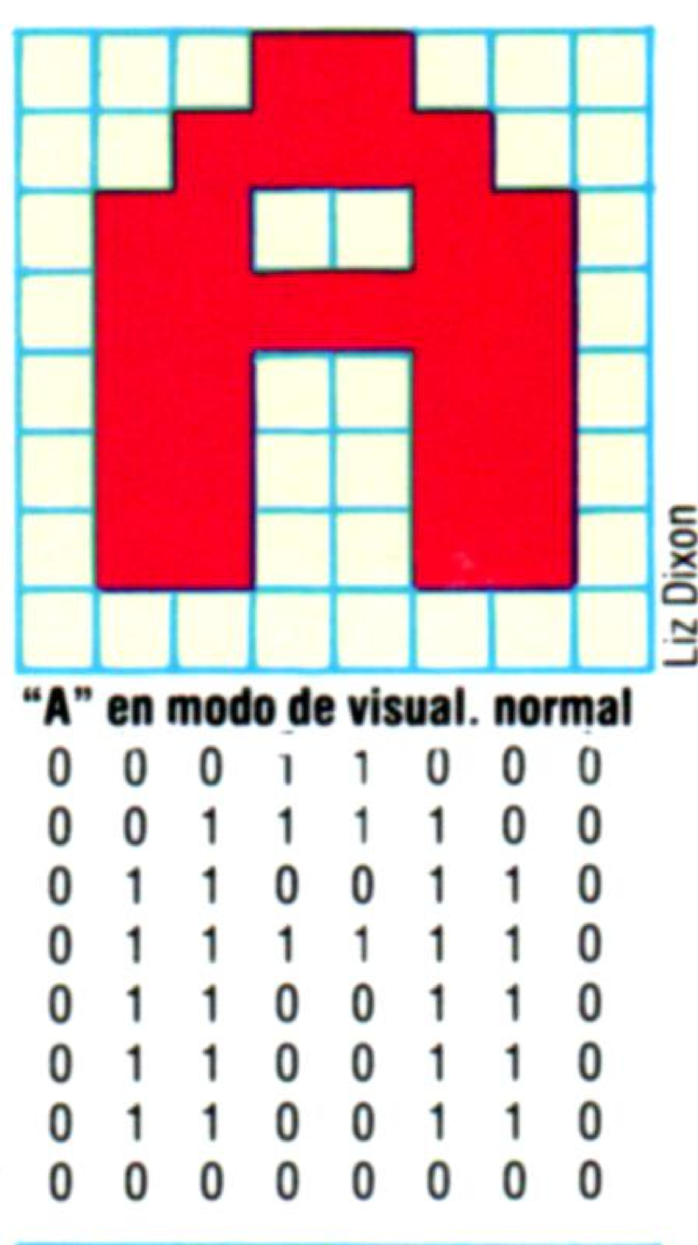
```
1070 REM ** SELECCION DEL DESPL. ALT. RES.
     /MAT. CAR DESDE BASE VENTANA **
1080 AD=4:REM SELECCIONA DESPLAZAMIENTO
     NORMAL
1090 POKE 53272,(PEEK(53272)AND240)
     OR2*AD
```

En principio, AD puede tomar valores entre 0 y 7, pero en la práctica existen limitaciones. Con VT, que vale 1 o 3, no podemos dar a AD el valor 2 o el 3, ya que éste es donde el VIC-II ve la imagen normal del carácter de ROM. Igualmente, un valor grande en AD pondrá la parte superior de la memoria en alta resolución fuera del alcance del VIC-II. La dirección correspondiente de la base de la actual pantalla en alta resolución o de matriz de caracteres se calcula así MC=VB+2048*AD (base ventana más desplazamiento), o bien así:

```
MC=16384*(3-(PEEK(56576)AND3))+1024*
(PEEK(53272)AND14)
```

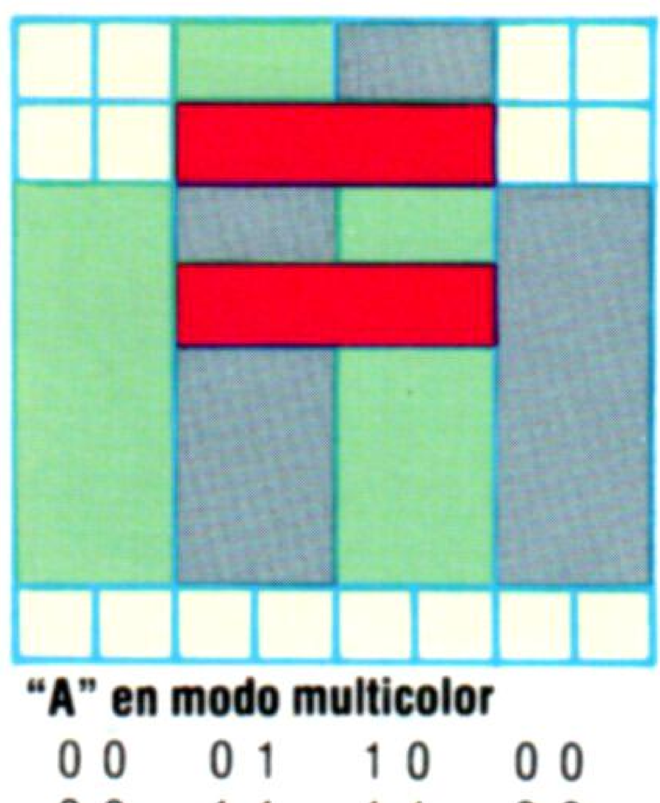
Activando estos registros podemos mover toda la memoria de visualización en video de un sitio a otro siempre que lo requiera el programa.

Colores coordinados
En el modo de visualización normal del Commodore 64, todo bit puesto a uno en los ocho bytes que definen un carácter se visualiza según el color actual de primer plano. Los bits con valor 0, por el contrario, se visualizan con el color del fondo de la pantalla. Al seleccionar el modo multicolor, los modelos de bits dejan de ser interpretados como bits individuales para serlo por pares. Las 4 combinaciones posibles de un par de bits representan los colores de primer plano y de fondo así como dos multicolores extra



"A" en modo de visual. normal

0	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0
0	1	1	1	1	1	1	0
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0
0	0	0	0	0	0	0	0

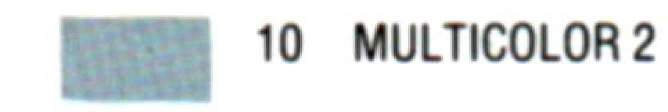


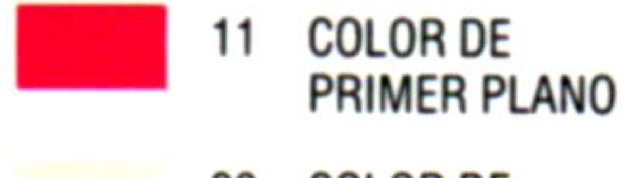
"A" en modo multicolor

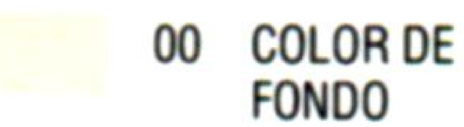
00	01	10	00
00	11	11	00
01	10	01	10
01	11	11	10
01	10	01	10
01	10	01	10
01	10	01	10
00	00	00	00

Clave

 01 MULTICOLOR 1

 10 MULTICOLOR 2

 11 COLOR DE PRIMER PLANO

 00 COLOR DE FONDO

Al salir la luna

En "Knight Lore" (El caballero Lore), juego que cuenta con unos gráficos tridimensionales excelentes, el héroe se transforma en hombre lobo



La guarida del licántropo
El objetivo de *Knight Lore* consiste en llevar a la caldera el objeto necesario para impedir que el caballero se convierta en un hombre lobo. Para hacer esto, primero es preciso visitar al brujo para descubrir qué necesitará. Entonces puede iniciarse la búsqueda en serio. Aunque los objetos pueden cambiar de posición, siempre se los encontrará en las mismas habitaciones. Por ejemplo, para llegar a la bota uno debe subir las mesas a la columna y trepar. Por otra parte, para llegar a la copa hay que esperar a que se abra la puerta correcta

Ian McKinnell

Ultimate siempre se ha destacado por producir juegos de calidad para el Sinclair Spectrum. La empresa ha creado una larga lista de juegos de laberinto en los cuales el jugador debe luchar para abrirse paso a través de las extrañas y maravillosas criaturas que habitan las diversas habitaciones. Mientras tanto, el jugador también debe recoger tesoros, así como talismanes y otras bagatelas místicas que, una vez unidas, completarán el juego. Ejemplos de este tipo son *Atic atac* y *Sabre Wulf*.

A pesar de la popularidad de estos juegos entre los usuarios del Spectrum, ahora la empresa se ha desmarcado de la aventura recreativa bidimensional tipificada por *Sabre Wulf*. Aunque guarda algunas semejanzas con las anteriores creaciones de Ultimate, *Knight Lore* es un tipo de juego muy diferente. El argumento es el siguiente: un caballero (en realidad, Sabre Man, el héroe de varias aventuras anteriores) busca el castillo de un hechicero para descubrir el conjuro que impedirá que él se convierta para siempre en un hombre lobo. El jugador dispone de cuarenta días con sus respectivas noches para hallar la respuesta, cuyo transcurso se va visualizando en la parte inferior de la pantalla; en la esquina inferior derecha hay una ventana que contiene un "cuadrante solar". Durante el día el jugador es Sabre Man. Sin embargo, cuando en la ventana aparece la luna, se transforma, con espasmos en el mejor estilo de Lon Chaney, en el Hombre Lobo. Afortunadamente, ello no tiene ninguna influencia en la capacidad del jugador para desplazarse, a excepción del momento de la transformación propiamente dicho, lo que puede ser un inconveniente si está intentando librarse de algún personaje monstruoso.

Al comenzar el juego, lo primero que usted percibirá es la incorporación de gráficos tridimensionales a la aventura. Estos gráficos son los mejores que se han visto hasta ahora para el Spectrum, incluyendo *Ant attack*. No obstante, a diferencia de este último, en *Knight Lore* no se puede cambiar el ángulo de visión, pero esto se compensa gracias a las ingeniosas formas en que se han construido las diversas habitaciones.

Knight Lore no es una aventura recreativa *per se*, sino más bien una serie de acertijos lógicos que se han de resolver. En este sentido, se acerca al auténtico juego de aventuras, al plantear arduos problemas que deben resolverse para poder seguir avanzando. Aunque muchas de las habitaciones están vacías, otras le ofrecerán desafíos que deberá resolver para salir por el otro extremo de la habitación. Una de ellas, por ejemplo, está habitada por pequeñas criaturas con apariencia de fantasmas que corren por el suelo: si consiguen tocar a Sabre Man, pierde una vida.

Los enigmas presentan diversos niveles de dificultad. Algunos son comparativamente fáciles, otros son de una complejidad diabólica, y varios sólo constituyen pistas falsas. Usted puede, por ejemplo, llegar a una habitación con bloques, encima de los cuales hay grandes globos con púas. Si intenta cruzar los bloques, estas esferas punzantes simplemente pincharán su entusiasmo; la respuesta no es otra que ¡pasar caminando por entre los bloques! Otro truco que utilizan los programadores es valerse de una representación bidimensional de un espacio tridimensional. Con frecuencia, bloques que parecen estar sobre el suelo en realidad están flotando en el aire, y si Sabre Man cae debajo de uno de ellos, no tendrá forma alguna de salir como no sea perdiendo una vida entre las púas e intentando volver a cruzar la habitación.

En muchas de las habitaciones hay objetos a recoger, tales como piedras preciosas, cálices y pociones. Una vez se llega a éstos, para cogerlos hay que saltar encima del objeto y tirar de la palanca de mando hacia atrás. Con ello el objeto quedará incluido en el "inventario" del jugador, que se visualiza en la esquina inferior izquierda de la pantalla. Esta maniobra puede ser útil para ganar altura cuando hay que saltar una pared que, de otra forma, sería insuperable.

Sabre Man puede moverse en las cuatro direcciones, utilizando ya sea el teclado o bien la palanca de mando. Los saltos se consiguen pulsando las teclas de la tercera fila o bien el botón de disparo. La orientación correcta de Sabre Man es esencial para desarrollar con éxito el juego; saltar en la dirección equivocada, por ejemplo, puede ser fatal. Quizá usted no tenga más remedio que pasar los primeros juegos simplemente familiarizándose con los mandos, para obtener un "ajuste exacto" de la orientación correcta, dado que movimientos muy ligeros de la palanca de mando pueden alterar la forma en que quede orientado Sabre Man.

Knight Lore es un juego fascinante y responde al elevado estándar de Ultimate. Incluso los jugadores a los que no les agrade encontrarse de vez en cuando con enigmas el juego les resultará sumamente absorbente, porque las soluciones a muchos de los problemas exigen respuestas rápidas con las teclas o el botón de disparo, además de exigir una mente aguda y atenta.

Knight Lore: Para el Spectrum de 48 K
Editado por: Ashby Ltd, Ashby de la Zouch, Leicestershire, LE6 5JU, Gran Bretaña
Autores: Ultimate Play the Game
Palanca de mando: Opcional
Formato: Cassette



Con este fascículo se han puesto a la venta las tapas correspondientes al séptimo volumen



El juego de tapas va acompañado de un sobre con los transferibles, numerados del 1 al 8, correspondientes a los volúmenes de que consta la obra; esto le permitirá marcar el lomo de cada uno de los volúmenes, a medida que aumente su colección.

Para encuadernar los 12 fascículos que componen un volumen, es preciso arrancar previamente las cubiertas de los mismos.

No olvide que, antes de colocar los fascículos en las tapas intercambiables, debe usted estampar el número en el lomo de las mismas, siguiendo las instrucciones que se dan a continuación:

1 Desprenda la hojita de protección y aplique el transferible en el lomo de la cubierta, haciendo coincidir los ángulos de referencia con los del recuadro del lomo.

2 Con un bolígrafo o un objeto de punta roma, repase varias veces el número, presionando como si quisiera borrarlo por completo.

3 Retire con cuidado y comprobará que el número ya está impreso en la cubierta. Cúbralo con la hojita de protección y repita la operación anterior con un objeto liso y redondeado, a fin de asegurar una perfecta y total adherencia.

Cada sobre de transferibles contiene una serie completa de números, del 1 al 8, para fijar a los lomos de los volúmenes. Ya que en cada volumen sólo aplicará el número correspondiente, puede utilizar los restantes para hacer una prueba preliminar.

PETICION DE FASCICULOS ATRASADOS

Si desea recibir algún fascículo atrasado o tapas, según las condiciones establecidas en el recuadro de la segunda página de cubierta («servicio de suscripciones y atrasados»), basta que rellene en LETRAS MAYUSCULAS este boletín y lo envíe a Editorial Delta, S.A., Paseo de Gracia, 88, 5.º - 08008 Barcelona.

NOMBRE
APELLIDOS
FECHA NACIMIENTO, DIA MES AÑO
PROFESION
DOMICILIO
N.º PISO ESCALERA
CODIGO POSTAL
POBLACION
PROVINCIA

OBRA:

N.ºs de fascículos atrasados que desea recibir: ....

....

N.º de tapas: ....

....

